AAMRL-TR-88-059

A KINEMATIC STUDY OF THE MERLIN 6500 ROBOT AND THE UTAH/MIT DEXTEROUS HAND AND A SIMULATION OF THEIR COMBINED BEHAVIOR

Ranvir Singh Solanki
Kuldip S. Rattan
Wright State University
3640 Colonel Glenn Highway
Fairborn, OH 45324

September 1988

Final Report for the Period April 1987 to September 1988

HARRY G. ARMSTRONG AEROSPACE MEDICAL RESEARCH LABORATORY
HUMAN SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6573

NOTICES

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Please do not request copies of this report from the Armstrong Aerospace Medical Research Laboratory. Additional copies may be purchased from:

National Technical Information Service
5285 Port Royal Road
Springfield, Virginia 22161

Federal Government agencies and their contractors registered with Defense Technical Information Center should direct requests for copies of this report to:

Defense Technical Information Center
Cameron Station
Alexandria, Virginia 22314

TECHNICAL REVIEW AND APPROVAL

AAMRL-TR-88-059

This report has been reviewed by the Office of Public Affairs (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

JAMES W. BRINKLEY
Acting Director
Biodynamics & Bioengineering Division
Harry G. Armstrong Aerospace Medical Research Laboratory

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

ADA203907

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AAMRL-TR-88-059	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION AAMRL/BBM 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB OH 45433-6573	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 61102F; PROJECT NO. 2312; TASK NO. V6; WORK UNIT ACCESSION NO. 02

11. TITLE (Include Security Classification)
A KINEMATIC STUDY OF THE MERLIN 6500 ROBOT AND THE UTAH/MIT DEXTEROUS HAND AND A SIMULATION OF THEIR COMBINED BEHAVIOR (UNCLASSIFIED)

12. PERSONAL AUTHOR(S)
Ranvir Singh Solanki and Kuldip S. Rattan, Wright State University

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Summary	FROM Apr 87 TO Sep 88	1988, September	103

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Robotics, Kinematics
05	08		
06	02		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report deals with the kinematics of the advanced, dexterous, four-fingered, sixteen jointed end-effector system called the UTAH/MIT hand and the industrial type six jointed MERLIN 6500 manipulator. The methodology of kinematic analysis, the direct and inverse kinematics of the MERLIN manipulator and the direct kinematics of the UTAH/MIT hand are presented. A computer graphical simulation program for the two systems, when combined together, is also carried out in this study. Certain key issues involved in the development of kinematics for manipulator systems with dexterous end-effectors are also discussed. (KF)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Ints Kaleps	22b. TELEPHONE (Include Area Code) (513) 255-3665	22c. OFFICE SYMBOL AAMRL/BBM

DD Form 1473, JUN 86 *Previous editions are obsolete.*

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

ACKNOWLEDGEMENT

This project was supported by the Modeling and Analysis Branch of the Armstrong Aerospace Medical Research Laboratory (AAMRL), Wright - Patterson Air Force Base, Ohio.

The authors would like to acknowledge Dr. Ints Kaleps, Dr. Daniel W. Repperger, Capt. Ronald Julian, Capt. Terrel Scoggins and Capt. Mark Jaster of AAMRL for their suggestions and help during the course of this study.

Mr. Jack Coate greatly assisted in the development of the graphics simulation, for which the authors would like to express their appreciation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I INTRODUCTION

An Overview

The Air Force has a need to maintain force survivability and base operability during wartime scenarios in chemical, biological and radiological environments. The Robotic Telepresence program at the Armstrong Aerospace Medical Research Laboratory (AAMRL) at Wright Patterson Air Force Base, Ohio, is based on the need to project human intelligence, perceptual capabilities, and motor skills into hostile environments through the use of human driven robotic systems, thereby removing humans from the hazardous environment. The Robotic Telepresence program at AAMRL investigates the feasibility of utilizing remote human-in-the-loop control of mobile dexterous robots to perform tasks such as aircraft inspection and servicing, explosive ordinance disposal, and environmental monitoring and decontamination.

The Robotic Telepresence concept projects human judgment, dexterity and adaptability in real time into a lethal environment. The program at AAMRL will develop a series of dynamic telepresence test cells incorporating driving systems attached to the human arm and hand as well as remote driven systems involving manipulators and dexterous end-effectors, amongst other state-of-the-art components. The remote system currently being evaluated consists of the Utah/MIT dexterous hands as suitable end-effectors to be attached to the end of robot arms such as the Merlin 6500 manipulator.

The task of integrating a system like the Utah/MIT dexterous hand to a robot arm is kinematically complex, especially in light of the fact that these two systems are a major part of a remote teleoperation system. The Merlin robot arm is kinematically unlike the human arm while the Utah/MIT hand differs from the human hand in some aspects, including the positioning of the thumb and the number of digits. This makes the task of comparison between the human system and the slave systems a difficult process at best. The integration task is further complicated by the presence of the remotizer, which performs the function of locating the actuators of the Utah/MIT dexterous hand away from the physical hand itself. These complications, amongst others, result in the need for a complete kinematic understanding of the Merlin

manipulator as well as the dexterous hand, as well as a means for the depiction and determination of possible complications that may arise when the two slave sub-systems are attached together.

As a first step in aiding the AAMRL in this research task, the authors have performed a complete kinematic study of the Merlin 6500 robot arm and the direct kinematics of the UTAH/MIT dexterous hand. This study has been performed with the basic fact in mind that a human arm will be involved in the feedback loop and will be directing the robot/dexterous hand combination in performing tasks with the teleoperated system. Further, to study the problem of attachment of the Utah/MIT hand to the Merlin, the authors have developed a computer graphical simulation program that allows a user to study different attachment schemes and the effect that these schemes may have on the kinematic behavior of the slave system.

The Objective

The objective of this project is to develop the closed-form forward and inverse kinematic solutions of the Merlin 6500 robot arm and the closed-form forward kinematic solution of the UTAH/MIT dexterous hand. A computer graphical simulation of the two systems, when connected together in user-defined configurations, is also performed in this study. The aim of the computer graphic simulation is to visually depict the effect of different attachment schemes on the kinematic behavior of the slave sub-system when combined together in specific user-specified configurations, and to prepare the ground-work for future research with the remote teleoperated systems.

Some Important Factors

One factor kept in mind during the development of computer simulations was the need for the software to be transportable to the sponsor's systems. A second factor was the need to allow for changes when adapting the simulation to the sponsor's available graphics packages. As a result, all source code was written in a modular fashion in a commonly used language (FORTRAN), and utilizes as few routines out of the graphics package (DISSPLA) as possible. The simulations are user-friendly and provide for modification and development as the

teleoperation study proceeds. An important factor considered during the development of the kinematic equations was that the robot/hand combination would be driven from a remote location by a human arm encased in an exo-skeleton. This resulted in the kinematics study being performed using kinematic frames that could be compared to a human arm/hand system.

II BACKGROUND

The Need for a Teleoperative System

Since the beginning of the present decade, it has been found necessary to perform manipulations in environments unsuitable for the presence of a human being. Some such hazardous environments include radiation hazard zones, chemical or biological hazard areas, undersea, deep space, etc. A human being would find it extremely hazardous, if not impossible, to exist in such environments, and since it is necessary to project human judgement and adaptability to perform unstructured tasks which require dexterous manipulation, there exists a need for remotely operated dexterous systems which provide a means for projecting human cognitive and motor functions into such environments. Such systems, when fully developed, will allow an operator, present at a comparatively safe location, to perceive and perform manipulation tasks just as if the operator was physically present at the remote work site.

Current Work

To achieve the above objective, various teleoperation systems dedicated to performing tasks in specific hostile environments have been developed over the last twenty years. Under-sea teleoperative systems have been successfully used to perform dexterous manipulations. The recent Titanic exploration performed by the Woods Hole Oceanographic Institute using the manned submersible "Alvin" and the tethered manipulator, "Jason, Jr"[1] is one example. Other efforts include the Advanced Integrated Manipulation System (AIMS - a prototype remote-handling system for use in hazardous environments developed by the Oak Ridge National Laboratory) where the master arms are kinematic replicas of the slave arms[2]; the U.S. Army's Human Engineering Laboratory Soldier Robot Interface Project (SRIP)[2], meant for battlefield scenarios; the ORNL/NASA Man-Equivalent Tele-Robot (METR), which is a modularized seven degree-of-freedom manipulator[2]; the Remote Operations and Maintenance Demonstration (ROMD) consisting of the model M_2 manipulator (a dual-arm force-reflecting bilateral servo-manipulator system)[2]; the Marine Corps/Naval Ocean Systems Center's (NOSC) Ground-Air TEleRobotics Systems (GATERS)[2]; and the current work being

done in other countries with multiple prehension manipulator systems in tele-robotic applications[3,4]. All these teleoperative systems have been designed to be able to perform dexterous manipulation tasks in specific environments.

The UTAH/MIT Dexterous Hand

The development of anthropomorphous systems for bio-engineering applications has resulted in research efforts being directed towards the implementation of dexterous systems which could be utilized for a large variety of manipulation tasks. The existence of a naturally occurring, highly complex system like the human hand has led to the development of semi-anthropomorphic, dexterous manipulator end-effectors. One such system in current existence is the UTAH/MIT dexterous hand, a sixteen degree-of-freedom system consisting of three four-jointed fingers and a four-jointed thumb situated off-center in the palm. A left- and right-pair of these hands will be used at AAMRL as a research testbed to experimentally investigate the various issues associated with human-in-the-loop control of dexterous end-effectors.

The Merlin 6500 Robot

The Utah/MIT dexterous hand will be attached to the Merlin 6500 six degree-of-freedom robot arm to form the remote manipulator system. Considerations such as the payload capacity, maximum tool-tip speed, accuracy and repeatability when encumbered by the heavy dexterous hand/remotizer system, the primary cost, and the availability of sufficient degrees of freedom to allow dexterous operation, etc. affected the choice of the manipulator for the purpose of evaluating the feasibility of integrating the Utah/MIT hands to a robot arm.

The Merlin 6500 robot arm is a six degree-of-freedom industrial manipulator with a payload capacity of 50 lbs. and a reach of 40 inches. The repeatability of the Merlin 6500 arm is ±0.001 inches. Each of the six degrees-of-freedom, viz. the waist, shoulder, elbow, wrist roll, wrist pitch and hand roll are controllable through a digital computer. The Merlin arm is therefore well suited to the task of moving around in three-dimensional space with the Utah/MIT hands attached at the end.

Issues

The first issue that arises when linking a robot arm to an end-effector system is the fact that the two systems have to be combined together physically to be able to perform a set of tasks. The next issue that must be addressed is the availability of a suitable work-space provided by the combined systems, such that performance of the desired tasks when combined together would not be inhibited. Both issues require the robot and end-effector to be kinematically understood and accurately modeled.

The actual combination of the Utah / MIT dexterous hand to the Merlin robot arm is complicated by the presence of a "remotizer", a multiple-bar linkage mechanism that allows the pneumatic actuation system of the Utah/MIT hand to be located away from the physical hand. This study, however, does not deal with the remotizer in any way beyond the acknowledgement of its presence as a constraint in the achievement of anthropomorphic arm geometry as regards the robot/dexterous hand combination. This study is intimately concerned with the kinematics of the two remote systems, specifically, with the direct and inverse kinematics of the Merlin 6500 robot arm and the direct kinematics of the Utah/MIT dexterous hand.

It is necessary to study the best possible method of attachment of the Utah/MIT hand to the Merlin robot arm. It is also needed to study the behaviour of the two systems when combined together and to obtain an idea of the attachment component for the two systems. This can either be done using actual models of the robot arm, the dexterous hand and suitable attachment pieces, or can be performed using a computer graphical simulation, or both. The computer graphical simulation method offers the advantage of being less costly and allows for many more possible kinematic attachment methods to be studied. The simulation can also be used to study the movement of the manipulator and end-effector, as well as assist in modeling the system's kinematic behaviour. With this fact in mind, a computer graphic simulation has been developed to model the behaviour of the left-shouldered Merlin arm and the left-fingered UTAH/MIT dexterous hand. A minimum set of commands from the graphics package (DISSPLA) have been used to allow for transportability of the software to the sponsor's site.

III SPATIAL TRANSFORMATIONS

Descriptions

Robotic manipulation requires that the end-effector be moved around in space. This involves describing positions and orientations of the mechanism in a mathematical form. The definition of manipulator position and orientation and the manipulation of mathematical quantities which represent position and orientation is performed by using coordinate systems (or frames) and transformations, which contain the description of both positions and orientations.

Description of a Position

The position of any point P in the universe can be represented with respect to a base frame by a [3x1] position vector. As different coordinate systems can be used, vectors must be tagged with information identifying which coordinate system they are described in. A leading superscript for a vector indicates the coordinate system in which it is referenced, for example, ${}^{A}P$ refers to the position of point P, which is described by three numerical values indicating distances along the axes of frame {A}. Individual components of a vector are identified by the trailing subscript x, y and z. Thus, the positional representation of point P relative to {A} would be written as

$$ {}^{A}P = \begin{bmatrix} {}^{A}P_x & {}^{A}P_y & {}^{A}P_z \end{bmatrix}^T \qquad (3.1) $$

where T denotes the transpose of the matrix.

Description of an Orientation

The complete location of a body in space is not specified until its orientation is also given. A point on a body could be oriented arbitrarily while being at the same position with respect to the base frame. To describe the orientation of a body, we attach a coordinate system {B} to the body and then give a description of the coordinate system relative to the reference system {A}.

Thus, positions of points are described with [3x1] vectors, while orientations of bodies are described by body-attached coordinate systems. One convenient way to describe the body-attached coordinate system is to describe the unit vectors of its three principal (orthonormal) axes in terms of the unit vectors in the universe (or base) coordinate system. It must be noted here that the description of two vectors would suffice, since the third can be obtained by taking the cross-product of the given two. The unit vectors along the principal directions of the body-attached coordinate frame {B} can be denoted as X_B, Y_B and Z_B. When written in terms of the universe or base coordinate system {A}, these vectors are written as ${}^{A}X_B$, ${}^{A}Y_B$, and ${}^{A}Z_B$. It is convenient to stack these unit vectors together as the columns of a [3x3] matrix, in the order ${}^{A}X_B$, ${}^{A}Y_B$ and ${}^{A}Z_B$. This [3x3] matrix is the rotation matrix which describes {B} relative to {A} and is written as ${}^{A}_{B}R$. Explicitly, ${}^{A}_{B}R$ is given by

$$ {}^{A}_{B}R = \begin{bmatrix} {}^{A}X_B & {}^{A}Y_B & {}^{A}Z_B \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad (3.2) $$

Description of a Transformation

The information needed to completely specify the whereabouts of the manipulator end-effector is its position and orientation. The point on the body whose position is chosen to be described is the origin of the body-attached frame {B}. The position and orientation pair which completely describes a body's whereabouts is combined together to form a transformation, which is defined as a set of four vectors giving position and orientation information. It must be remembered here that a frame is an orthogonal coordinate system which is described relative to some other frame. Thus, when the frame {B} is described with respect to the frame {A}, then ${}^{A}_{B}T$ can be represented as

$$ {}^{A}_{B}T = \begin{bmatrix} {}^{A}_{B}R, & {}^{A}P_{B_{org}} \end{bmatrix} \qquad (3.3) $$

where ${}^{A}_{B}R$ is the rotation matrix representation of {B} relative to {A}, and is specified by equation (3.2), and ${}^{A}P_{B_{org}}$ is the vector from the origin of {A} to the origin of {B} and can be written according to equation (3.1).

Transformations: Changing Descriptions from Frame to Frame

In robotic kinematics, we are concerned with describing position and orientation in various reference coordinate systems. Thus, we need to be able to transform this information from frame to frame rather frequently.

Translated Frames

Let the position of the point P be defined with reference to the frame {B} as shown in Figure 1. It is required to express the position of P with respect to {A}. When {A} has the same orientation as {B}, the difference in {A} and {B} can be represented by a translation vector, ${}^{A}P_{B_{org}}$, which locates the origin of {B} with respect to {A}.

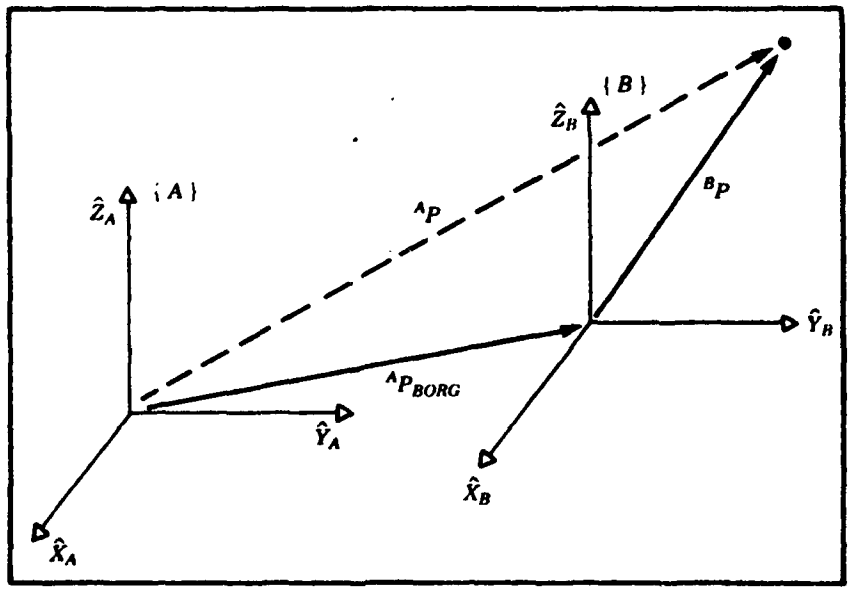


Figure 1. Translational Mapping.[5]
{B} has the same orientation as {A}.

Since both vectors are defined relative to frames of the same orientation, we can compute the description of point P relative to {A} by the use of vector addition:

$$^{A}P = {}^{B}P + {}^{A}P_{B_{org}} \tag{3.4}$$

It must be remembered that it is possible to add vectors that are defined in terms of different frames only if the frames have the same orientation. It must also be noted here that the point P has itself not moved in space - only its description has changed.

The vector $^{A}P_{B_{org}}$ defines a translational mapping of point P from its description in {B} to {A}, since all the information needed to perform the change in description is contained in $^{A}P_{B_{org}}$ (along with the knowledge of their equivalent orientation).

Rotated Frames

The matrix $^{A}_{B}R$ describes the relative orientation of {B} with {A} and is composed of the three column vectors, $^{A}X_{B}$, $^{A}Y_{B}$ and $^{A}Z_{B}$. By our definition, the columns of a rotation matrix have unit magnitude and represent vectors that are orthonormal. Since the inverse of a matrix with orthonormal columns is equal to its transpose, we have

$$^{A}_{B}R = {}^{B}_{A}R^{-1} = {}^{B}_{A}R^{T} \tag{3.5}$$

Thus, since the column vectors of $^{A}_{B}R$ are the unit vectors of {B} written in {A}, the rows of $^{A}_{B}R$ are the unit vectors of {A} written in {B}.

As such, a rotation matrix can be interpreted as a set of three column vectors or as a set of three row vectors as follows:

$$^{A}_{B}R = \begin{bmatrix} {}^{A}X_{B} & {}^{A}Y_{B} & {}^{A}Z_{B} \end{bmatrix} = \begin{bmatrix} {}^{B}X_{A}^{T} & {}^{B}Y_{A}^{T} & {}^{B}Z_{A}^{T} \end{bmatrix}^{T} \tag{3.6}$$

We often need to know the components of a vector with respect to a frame {A} when we know its components with respect to a frame {B}, where the origins of frame {A} and {B} are coincident (Figure 2). This computation is possible when a description of the orientation of {B} is known with respect to {A}. This orientation is given by the rotation matrix ${}^{A}_{B}R$.

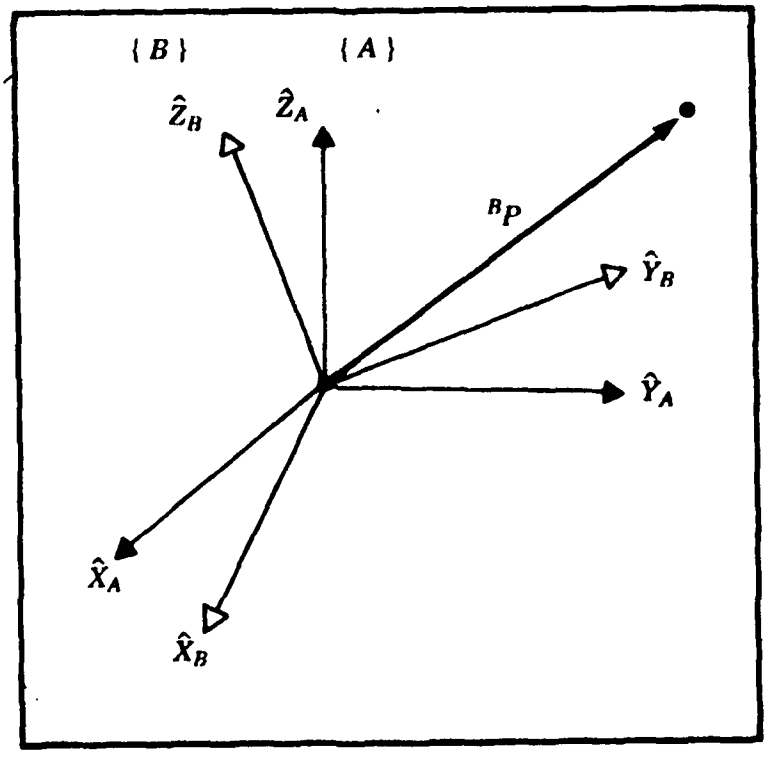


Figure 2. Rotated frames.[5]

Since the components of any vector are simply the projections of that vector onto the unit directions of its frame, the projection is computed by the vector dot product. Thus, the components of ${}^{A}P$ can be computed as

$$\begin{aligned} {}^{A}p_x &= {}^{B}X_A \cdot {}^{B}P, \\ {}^{A}p_y &= {}^{B}Y_A \cdot {}^{B}P, \\ {}^{A}p_z &= {}^{B}Z_A \cdot {}^{B}P. \end{aligned} \qquad (3.7)$$

In order to express the above equation in terms of a rotation

matrix multiplication, we note from the previous equation that the rows of ${}^{A}_{B}R$ are ${}^{B}X_{A}$, ${}^{B}Y_{A}$ and ${}^{B}Z_{A}$. As such, the above equation can be written compactly as

$$ {}^{A}P = {}^{A}_{B}R \; {}^{B}P \qquad (3.8) $$

Equation (3.8) implements a rotational mapping from frame {B} to frame {A}, i.e. it changes the description of a vector from ${}^{B}P$ into ${}^{A}P$.

Mappings Involving General Frames

We can now address the problem of mappings involving general frames, i.e. those frames where both translational and rotational differences are involved. In this case, the frames {A} and {B} do not have coincident origins, nor do they possess equivalent orientations. The vector that locates {B}'s origin relative to {A} is called ${}^{A}P_{B_{org}}$, while the rotation of frame {B} relative to {A} is given by ${}^{A}_{B}R$. Given ${}^{B}P$, the vector describing the point P with respect to frame {B}, we wish to compute ${}^{A}P$, the description of the vector relative to {A}.

This is done by changing ${}^{B}P$ to its description relative to an intermediate frame which has the same orientation as {A}, but whose origin is co-incident to {B}. This is mathematically performed by pre-multiplying ${}^{B}P$ by ${}^{A}_{B}R$, as seen previously in (3.8). We can now translate between origins by performing simple vector addition, since the intermediate frame and {A} have equivalent orientations. Mathematically speaking, this is done as follows:

$$ {}^{A}P = {}^{A}_{B}R \; {}^{B}P + {}^{A}P_{B_{org}} \qquad (3.9) $$

The above equation describes a general transformation of a vector from its description in one frame to its description in another.

Since we are also interested in a concise notation, the above equation can be written as

$$ {}^{A}P = {}^{A}_{B}T \; {}^{B}P \tag{3.10} $$

where the operator ${}^{A}_{B}T$ is defined by

$$ \left[\begin{array}{ccc|c} & {}^{A}_{B}R \; (3x3) & & {}^{A}P_{B_{org}} \; (3x1) \\ \hline 0 & 0 & 0 & 1 \end{array}\right]_{4x4} \tag{3.11} $$

and the ${}^{A}P$ and ${}^{B}P$ vectors are embedded in [4x1] matrices.

The 4 x 4 matrix in (3.11) is called the homogeneous transformation operator. This transformation matrix consists of the position and orientation sub-matrices and represents a description of the frame {B} relative to frame {A} as well as the transformation of a vector described in terms of frame {B} to its description in {A}.

The Mathematics of Transformation Operators

Before we proceed further, it is advisable to explain the two important mathematical operations in manipulator kinematics regarding the transformation operator ${}^{A}_{B}T$, viz. concatenation and inversion.

Multiple transformations are performed when there exist more than two frames and one of the frames, say {C} (or a vector represented by one of the frames), needs to be mapped to the first frame {A} through the second frame {B}. This situation is encountered when the available description of the frames includes a description of the third frame {C} relative to the second one {B}, i.e. ${}^{B}_{C}T$ is known, and the second frame {B} relative to the first frame {A}, i.e. ${}^{A}_{B}T$ is known. The compound transformation is mathematically performed by the use of matrix multiplication operations as follows

$$ {}^{A}_{C}T = {}^{A}_{B}T \; {}^{B}_{C}T \tag{3.12} $$

Here, ${}^{A}_{C}T$ represents the homogeneous transform or mapping of frame

{C} with respect to frame {A}. We notice the notational convenience here - the leading sub-script of the first term on the right side of the above equation may be said to "cancel" the leading super-script in the second term on the right side of the equation, to give the term on the left side of the equation.

In many cases, it is necessary to perform a transformation matrix inversion. Typically, this is done where the order of frame descriptions is found to be incompatible with compound transformation proceedures. In the example given above, if the description of {B} relative to {C}, i.e. ${}^{C}_{B}T$ was known, then the determination of ${}^{A}_{C}T$ could only be performed by inverting ${}^{C}_{B}T$ to obtain ${}^{B}_{C}T$, and using the above equation to determine ${}^{A}_{C}T$. Thus, (3.12) will now become

$$ {}^{A}_{C}T = {}^{A}_{B}T \cdot {}^{C}_{B}T^{-1} \qquad (3.13) $$

The inversion of the transformation matrix could be easily performed by the generalized matrix inversion method. A computationally faster method (involving a fewer number of operations) and one which utilizes the inherent structure (orthogonality) of the rotation matrix to advantage is explained below.

To find ${}^{B}_{A}T$, we must compute ${}^{B}_{A}R$ and ${}^{B}P_{A_{org}}$ from ${}^{A}_{B}R$ and ${}^{A}P_{B_{org}}$. From (3.5), we have

$$ {}^{B}_{A}R = {}^{A}_{B}R^{T} \qquad (3.14) $$

and so we change the description of ${}^{A}P_{B_{org}}$ into {B} using the transformation involving general frames, as

$$ {}^{B}({}^{A}P_{B_{org}}) = {}^{B}_{A}R \cdot {}^{A}P_{B_{org}} + {}^{B}P_{A_{org}} \qquad (3.15) $$

Since the left side of the above equation is necessarily zero, we have

$$
\begin{aligned}
{}^{B}P_{A_{org}} &= -{}^{B}_{A}R \cdot {}^{A}P_{B_{org}} \\
&= -{}^{A}_{B}R^{T} \cdot {}^{A}P_{B_{org}} \qquad (3.16)
\end{aligned}
$$

We can therefore write $\begin{bmatrix} {}^{A}_{B}T \end{bmatrix}^{-1}$, i.e., ${}^{B}_{A}T$ as:

$$
\begin{bmatrix} {}^{A}_{B}T \end{bmatrix}^{-1} = {}^{B}_{A}T = \left[\begin{array}{ccc|c} & {}^{A}_{B}R^{T} & & -{}^{A}_{B}R^{T} \cdot {}^{A}P_{B_{org}} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \qquad (3.17)
$$

We can thus perform the inversion operation on the transform matrix using (3.17).

IV MANIPULATOR KINEMATICS

Manipulator kinematics defines the geometrical properties of motion. The direct kinematics problem is defined as the determination of the end-effector position and orientation when the joint variables are known [Appendix 1.7], while the inverse kinematic problem is defined as the determination of the joint variables to achieve the desired position and orientation [Appendix 1.8]. We will first examine the generalized direct kinematic problem, followed by a study of certain important factors involved in the generalized inverse manipulator kinematics problem.

In order to deal with the complex geometry of a manipulator, frames are affixed to various parts of the mechanism. When the mechanism articulates, the relationship between the frames describes the kinematic behaviour of the manipulator.

Joint Description

A manipulator consists of a set of links connected together in an open chain by joints.

Types of Manipulator Joints

Manipulator links can be joined together by a variety of joint types. The commonly existing manipulator joint types[6] consist of :

Revolute joint, where the joint consists of a simple hinge, with the only possible relative motion between the paired members being a rotation about the joint axis. This is the most commonly used joint in manipulators.

Prismatic joint, where the joint consists of a sliding type mechanism, with no relative rotation occurring between the jointed members. The only possible relative motion is a pure (rectilinear) translation along the slide direction. This is the next most commonly used joint in manipulators.

Helical joint. These are rarely found in manipulators due to the difficulty in powering the joint. The effect of a helical joint is normally obtained by a special combination of the revolute and prismatic joints. The joint acts like a screw-and-nut arrangement. It can be

substituted for by a co-axial revolute and prismatic joint with a constant ratio of rotational to translational displacement.

Cylindrical joint, which is in effect a revolute joint without the end constraints, i.e., sliding takes place along the revolute axis. This joint is normally found in manipulators as a co-axial revolute and prismatic joint, with each joint independantly powered and controlled.

Spherical joint, which consists of a spherical ball and socket arrangement. The relative motion is spherical, resulting in all points remaining at a fixed distance from the center point of the joint. In manipulators, the effect of this joint is obtained by three non-coplanar independently powered revolute joints whose axes always intersect at a point.

Flat planar joint, which consists of two flat planes sliding and turning on each other. It can be kinematically constructed by two non-planar prismatic joints and a revolute joint perpendicular to the directions of both the prismatic joints.

Although other manipulator joints do exist, they are rarely used due to the associated problems in powering and controlling them.

In certain cases, as in some of those above, there exist manipulator joints with more than a single degree of freedom. These joints can be kinematically modelled as 'n' joints of one degree of freedom each, connected together with 'n-1' links of zero link length. As such, we will, without loss of generality, consider manipulator kinematics with joints having single degrees of freedom at each joint.

Significant Dimensions of Joints

Significant dimensions for joints consist of the **link offset** (d_i) and the **joint angle** (θ_i). Neighbouring links are joined together at any one joint, which has an axis of motion that is common to both the links connected at the joint. The distance along this common axes, from one link to the next, is called the **link offset** (d_i). The link offset d_i for joint i is thus the distance measured along the axis of joint i, from the intersection of the common perpendicular between the axes of joints i-1 and i, to the intersection of the common perpendicular

between the axes of joints i and i+1. The joint angle θ_i describes the amount of rotation about the common axes at the joint, between one link and its neighbour. This parameter is measured as the angle from the extension of the common perpendicular between the axes of joint i and i-1 to the common perpendicular between the axes i and i+1, in a plane perpendicular to the axis of joint i. The link offset is considered to be the joint variable if the joint under consideration is prismatic in nature, while the joint angle is the joint variable if the joint under consideration is revolute.

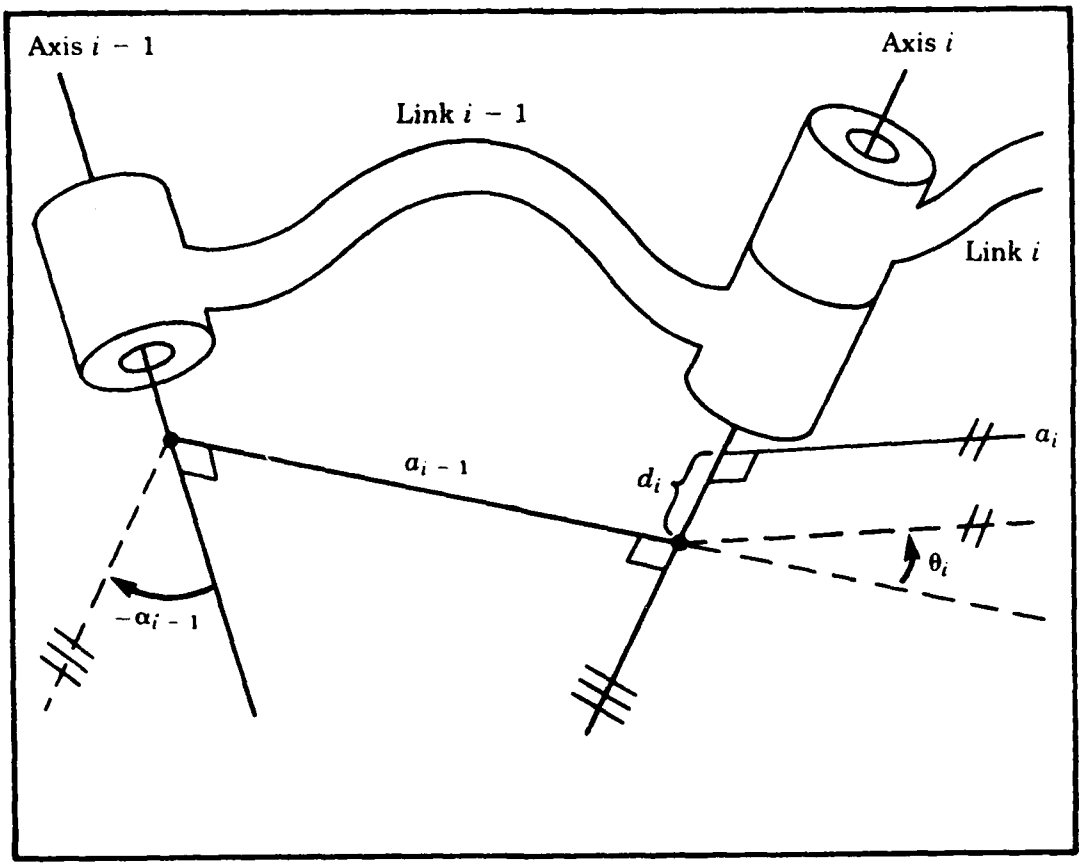


Figure 3. Link and joint parameters.[5]

LINK DESCRIPTION

We now examine the significance and kinematic representation of links as well as their description.

The Significance and Kinematic Representation of Links

Links are used to connect joints. The kinematic significance of links is that they maintain fixed configurations between their joints and other points and lines along the axis of the joints. It is important to note here that, regardless of the actual location, shape or size of a link, a manipulator may be completely represented kinematically by a skeleton diagram, which is a line drawing representation of the links of the manipulator.

Significant Link Dimensions

Significant dimensions of a link consist of the **link length** and the **link twist**. For any two joint axes in three dimensional space, there exists a well-defined measure of distance between them. The distance measured along a line which is mutually perpendicular to both axes defines the **link length**. The **link twist** is measured in a plane whose normal is the mutually perpendicular line between the two axes (the axes under consideration and the preceeding joint axes) and is defined by the angle formed between the projections on this plane of the two joint axes (see figure 3).

Any open kinematic chain can be described by specifying the values of the **joint angle**, **link offset**, **link length** and **link twist** for each joint-link system. Of these four parameters, three are constant for a joint, while the fourth parameter forms the joint variable. The specification of an open kinematic chain by means of these four quantities is known as the **Denavit-Hartenburg** convention[5,7].

The Denavit-Hartenburg Notation

The Denavit-Hartenburg notational convention involves the description of a robot arm by means of the **link length** a_{i-1}, **link twist angle** α_{i-1}, **link offset** d_i, and the **joint angle** θ_i. The method depends on the fixing of a frame to each joint of the robot and determining the joint parameters and joint variable range. We utilise the convention that frame {i} has its origin at joint axis i and is attached to link i. Thus, the parameter **link length** (a_{i-1}) is measured as the signed distance along the common perpendicular to the axes i-1 and i, from

joint axis i-1 to joint axis i. The **link twist** (α_{i-1}) is measured as the signed angle (using the right-hand rule) between the projection of axis i-1 to axis i on a plane whose normal is the mutually perpendicular line between axes i-1 and i. The **link offset** (d_i) is the signed distance measured along the axis of joint i from the point where a_{i-1} intersects the axis i, to the point where a_i intersects that axis. The **joint angle** (θ_i) is measured as the signed angle (using the right hand rule) between the extension of a_{i-1} and a_i, about the axis of joint i. In the special case of the joint being the first one under consideration, i.e. i is 1, the link parameters are determined from the base frame, here (i-1) is 0. Since link length a_i and link twist α_i depend on joint axes i and i+1, the parameters at the end of the chain, a_n and α_n, are set to 0 and do not need to be defined.

Affixing Frames to Links

In order to describe the location of each link relative to its neighbours, a **frame** is attached to each link. The link frames are named according to the link to which they are attached, i.e. frame {i} is rigidly attached to link i.

The convention adopted for affixing frames to links depends on whether the link is an intermediate link or the first/last link in the chain.

First and Last Links in the Chain

We attach the frame {0} to the base of the robot, or to a non-moving section of the arm, called link {0}. This base, or reference frame, can also be set up with its origin coinciding with frame {1} when the joint 1 variable is 0 (the generally preferred method). The Z-axis of frame {0} coincides with the Z-axis of frame {1}, and so do the X and Y axes. This ensures that $a_0 = 0.0$ and $\alpha_0 = 0.0$. Additionally, $d_1 = 0.0$ if joint 1 is revolute, while $\theta_1 = 0.0$ if joint 1 is prismatic. However, when the base or reference frame is not located to coincide

with frame {1}, $a_0 \neq 0.0$ and $\alpha_0 \neq 0.0$. In this case, it is not necessary that θ_1 be equal to 0.0. The base frame {0} is then set up for mere convenience.

For joint 'n' revolute, the direction of X_n is chosen so that it aligns with X_{n-1} when $\theta_n = 0.0$, and the origin of frame {N} is chosen so that $d_n = 0.0$. In cases where three axes intersect at a point, frame {N} is located at the point of intersection of the three axes. If joint 'n' is prismatic, the direction of X_n is chosen so that $\theta_n = 0.0$ and the origin of frame {N} is chosen at the intersection of X_{n-1} and joint axes 'n' when $d_n = 0.0$.

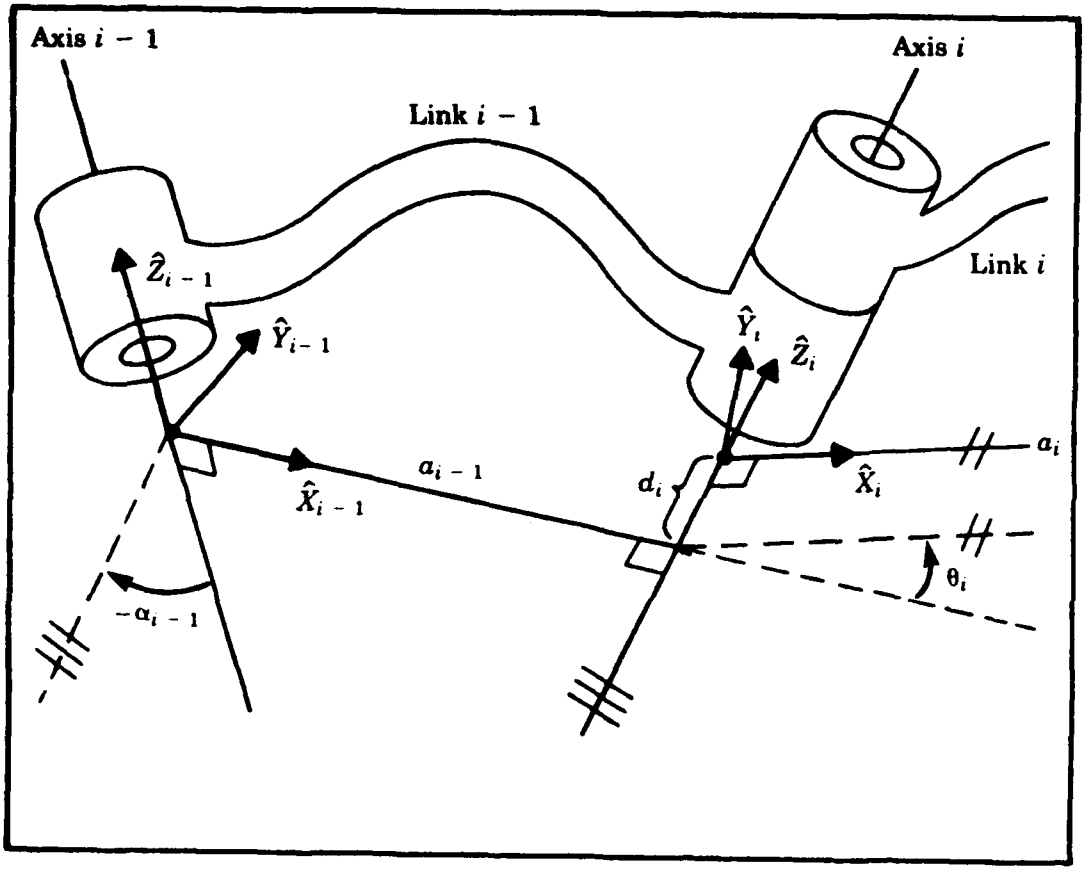


Figure 4. Link frames and kinematic parameters[5].

Intermediate Links in the Chain

The convention used to affix frames on intermediate links involves setting the Z-axis of frame {i}, called Z_i, coincident with the joint i

axis. The origin of frame {i} is located where the a_i perpendicular intersects the joint i axis. The direction of Z_i can be in either direction along the joint i axis. X_i is set up so that it points along a_i in the direction from joint i to joint i+1. In the special case of $a_i = 0$, X_i is chosen normal to plane of Z_i and Z_{i+1}. The **link twist** α_i is measured in the right hand sense about X_i. Y_i is formed by the right hand rule to complete the i^{th} frame. Figure 4 shows the location of the frames and the kinematic parameters.

The Link Parameters in Terms of the Link Frames

Attachment of the link frames to the links according to the convention described above results in the manipulator kinematic parameters being redefined in terms of the link frames as follows :

a_i = the signed distance from Z_i to Z_{i+1}, measured along X_i,

α_i = the signed angle between Z_i and Z_{i+1}, measured about X_i in the right hand sense,

d_i = the signed distance from X_{i-1} to X_i, measured along Z_i, and

θ_i = the signed angle between X_{i-1} and X_i, measured about Z_i in the right hand sense.

It must be noted here that the above convention does not result in a unique attachment of frames to links. When the Z_i-axis is aligned along joint axis i, there are two choices of direction in which to point Z_i. Also, in the case of intersecting joint axes (i.e. $a_i = 0$), there are two choices for the direction of X_i, corresponding to the choice of signs for the normal to the plane containing Z_i and Z_{i+1}.

Derivation of Link Transforms

The general form of the transformation which relates the frames attached to neighbouring links is now derived. These transformations

are then concatenated to solve for the position and orientation of link 'n' relative to link 0.

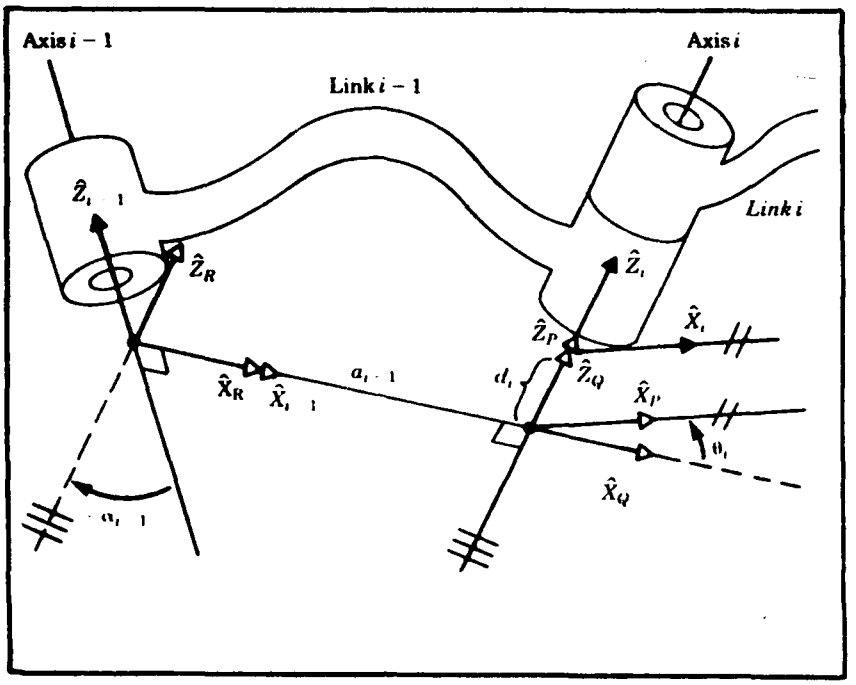


Figure 5. Intermediate link frames and kinematic parameters[5].

The determination of the transformation which defines frame {i} relative to frame {i+1} is, in general, a function of the four link parameters. For any given robot arm, this transformation will be a function of only one variable, the other three being fixed. It must be remembered here that we are dealing with multiple degree-of-freedom joints as multiple joints with one degree of freedom and zero offsets each. By defining a frame for each link, the kinematic problem has been broken into 'n' sub-problems. To solve each of these sub-problems, it is further necessary to divide them further into four sub-subproblems. Each of the four sub-subproblems consists of a basic transformation which is a function of one link parameter and can be written by inspection.

It is necessary to define three intermediate frames {P}, {Q} and {R} for each link. Figure 5 shows the same pair of joints as figure 4, with the intermediate frames {P}, {Q} and {R} defined. For clarity,

only the X and Z axes are shown.

In figure 5, frame {R} differs from frame {i-1} only by a rotation of α_{i-1}. Frame {Q} differs from {R} by a translation a_{i-1}. Frame {P} differs from {Q} by a rotation θ_i, and frame {i} differs from {P} by a translation d_i. To write the transformation which transforms vectors defined in {i} to their description in {i-1}, we write

$$ {}^{i-1}P = {}^{i-1}_{R}T \; {}^{R}_{Q}T \; {}^{Q}_{P}T \; {}^{P}_{i}T \; {}^{i}P \tag{4.1} $$

or

$$ {}^{i-1}P = {}^{i-1}_{i}T \; {}^{i}P \tag{4.2} $$

where

$$ {}^{i-1}_{i}T = {}^{i-1}_{R}T \; {}^{R}_{Q}T \; {}^{Q}_{P}T \; {}^{P}_{i}T \tag{4.3} $$

Equation (4.3) may therefore be written as

$$ {}^{i-1}_{i}T = \mathrm{Rot}(X_i,\alpha_{i-1}) \; \mathrm{Trans}(X_i,a_{i-1}) \; \mathrm{Rot}(Z_i,\theta_i) \; \mathrm{Trans}(Z_i,d_i) \tag{4.4} $$

or

$$ {}^{i-1}_{i}T = \mathrm{Screw}(X_i,\, a_{i-1},\, \alpha_{i-1}) \; \mathrm{Screw}(Z_i,\, d_i,\, \theta_i) \tag{4.5} $$

where $\mathrm{Screw}(Q,r,\phi)$ stands for a translation along an axis Q by a distance r, and a rotation about the same axis by an angle ϕ.

The general form of the transformation of vectors defined in frame {i} to their description in frame {i-1}, i.e. ${}^{i-1}_{i}T$, is obtained from (4.5) (detailed in Appendix 2), and is given by

$$
{}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i\, ca_{i-1} & c\theta_i\, ca_{i-1} & -sa_{i-1} & -sa_{i-1}\, d_i \\ s\theta_i\, sa_{i-1} & c\theta_i\, sa_{i-1} & ca_{i-1} & ca_{i-1}\, d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.6)
$$

where

$$
\begin{aligned} c\theta_i &= \text{Cos } \theta_i \\ s\theta_i &= \text{Sin } \theta_i \\ ca_{i-1} &= \text{Cos } a_{i-1} \\ sa_{i-1} &= \text{Sin } a_{i-1} \end{aligned}
$$

The Direct Kinematics of Manipulators

Having derived the link frames and the corresponding link parameters, developing the direct kinematic equations is a straight-forward process. Using the values of the link parameters, the individual link transform matrices are computed. The manipulator arm kinematic transformation matrices are then multiplied together to find the single transform that relates frame {N} to frame {0}, as shown in equation (4.7).

$$
{}^{0}_{N}T = {}^{0}_{1}T\ {}^{1}_{2}T\ {}^{2}_{3}T \ldots {}^{N-1}_{N}T \qquad (4.7)
$$

This transformation will be a function of all 'n' joint variables. The kinematic parameters for joint 'i' are a_{i-1}, a_{i-1} and d_i as well as θ_i, the joint variable for a revolute joint. Each of these parameters, as well as the joint variable, have to be determined for each link of the manipulator.

The Inverse Kinematics of Manipulators

The inverse kinematics problem involves the determination of the joint angles of the manipulator which will achieve the desired position and orientation. This more difficult problem can be solved by various

methods, of which the prominent and easily programmable ones utilise either geometric or algebraic manipulations to obtain a set of solutions. One of the important factors that has to be taken into account consists of whether the defined (known) position is at the tip of the manipulator or at some other convenient point along the last axis. Another important factor to be taken into consideration is whether there exists a solution for the desired position and orientation. Further, the solution set may consist of one or more solutions which will allow the achievement of the desired position and orientation, and a choice between these solutions must be made.

Solvability

The problem of solving the kinematic equations of a manipulator to determine the joint angles is a non-linear one. Given the values of each of the terms in ${}^{0}_{N}T$, we have to determine a viable set of joint angles θ_1, θ_2, θ_3, . . . ,θ_n. For a six degree-of-freedom arm, the set of joint angles that needs to be determined (the unknowns) is six. We have a total of 16 values obtained from the ${}^{0}_{N}T$ matrix, four of which (the last row) are trivial (equating 0 or 1 on both sides). Out of the remaining twelve known equations, three equations define the position values and are independant. From the nine remaining equations that arise from the rotation matrix part of ${}^{0}_{N}T$, only three equations are independant. These three equations, added with the set of three equations that arise from the position vector part of the transformation matrix, provide a set of six equations. For a six degree of freedom manipulator, we have six joint angles to be determined, and six equations. These equations are a set of non-linear transcendental equations which can be difficult to solve, specially for a general mechanism with six degrees of freedom with all link parameters non-zero. This is unlike industrial manipulators where the link parameters consist of twist angles of 0° or 90°, resulting in their sine and cosine values being 'nice' numbers like 0 or 1, or where many of the offsets are 0. As with any set of non-linear equations, it is necessary to look for the existence of solutions, multiple solutions and the method of solution.

The Existence of Solutions and Manipulator Workspaces

The question of whether or not there exists an inverse kinematic solution for the successful achievement of the desired position and orientation raises the question of manipulator workspace. Broadly speaking, workspace is that volume of three dimensional space which the end-effector of the manipulator can reach. For a solution to exist, the desired goal point must lie on or within the workspace boundaries. The **dexterous workspace** is that volume of space which the robot end-effector can reach with all orientations, i.e. at each point in the dexterous workspace, the end-effector can be arbitrarily oriented. The **reachable workspace** is that volume of space which the robot can reach in at least one orientation. Thus, the dexterous work-space of a robot is a sub-set of it's reachable workspace.

For each manipulator, there exists an outer and inner workspace boundary. Thus, there exists an outer reachable and an outer dexterous workspace boundary, as well as an inner reachable and inner dexterous boundary. The outer and inner workspaces are a function of the kinematic parameters of the manipulator and the joint variable range limits.

The Existence of Multiple Solutions

Another common problem encountered in solving manipulator inverse kinematic equations is that of multiple solutions. The existence of multiple solutions arises due to the kinematic arrangement of consecutive joints and the range of motion of each joint. For example, when there exist two joints with successive parallel horizontal axes, one of the ways to achieve the desired position is with the first link pointing upwards with the second link pointing downwards, while the same position is achievable by the first link pointing downwards and the second link pointing upwards. Another example of the existence of multiple solutions involves the orienting mechanism of the robot. For each solution, provided the joint variable ranges are not exceeded, there will exist a wrist 'flipped' solution. Also, the more nonzero link parameters there exist for the manipulator arm, the more ways there will be to achieve the desired goal. For a manipulator with six rotational joints, the maximum number of solutions is related to the

number of the link length parameters (a_i) that are equal to zero. The more that are nonzero, the bigger the number of solutions. For a completely general rotary-jointed manipulator with six degrees of freedom, there are up to 16 solutions that are possible. Table 1 shows the relationship between the link length parameters (a_i) and the number of solutions for a six degree of freedom manipulator.

Table 1. Number of Solutions vs. Nonzero a_i.[5]

a_i	Number of solutions
$a_1 = a_3 = a_5 = 0$	≤ 4
$a_3 = a_5 = 0$	≤ 8
$a_3 = 0$	≤ 16
All $a_i = 0$	≤ 16

The Methods of Solution

Unlike the process of solving a system of linear equations, there are no general algorithms that can be adopted to solve a set of non-linear equations. It therefore becomes necessary to note that a manipulator is considered solvable if the joint variables can be determined by an algorithm which allows the determination of all the sets of joint variables associated with the goal frames position and orientation.

The broad division of manipulator solution strategies is divisible into closed-form solutions and numerical solutions. Due to the iterative nature of numerical solutions, they are much slower in "solving" the manipulator than closed form solution techniques. Further, most numerical iterative techniques utilised in "solving" manipulators do not guarantee the finding of all possible solutions that may exist for the manipulator. Closed form methods involve a solution

based on analytical expressions or on the solution of a polynomial of degree 4 or less, such that non-iterative calculations suffice to arrive at a solution.

Within the class of closed-form solution techniques, two major distinctions can be made. The two sub-classes of the closed-form method include the purely algebraic solution process and the geometric process. The geometric process, however, does involve a degree of algebraic manipulation.

A recent major result is that all systems with revolute and prismatic joints having a total of six degrees of freedom in a single series chain are solvable, at least numerically. It is, however, true that it is only in special cases that robots with six degrees of freedom can be analytically solved. These robots possess the common characteristic of several intersecting joint axes and/or many α_i (twist angle) equal to 0° or $\pm 90^\circ$. A sufficient condition that a manipulator with six revolute joints will have a closed-form solution is that three neighbouring joint axes intersect at a point.

A well-known solution method for a manipulator with all six revolute joints and with three axes intersecting at a point is the Pieper's solution process. This consists of transferring the known position information about the goal point to the point of intersection of the three axes. Successive algebraic manipulations then leads to a solution. The advantage of Pieper's technique is the determination of kinematic singularities during the solution process, as well as the determination of all possible solutions to the inverse kinematics problem for the manipulator under consideration.

The geometric technique of closed-form solutions to inverse kinematics has never proven to be popular, due to its inherent disadvantage of not being able to provide kinematic singularity information. It does possess the advantage of providing information about the determination of which of the solutions is to be adopted. However, the technique works to advantage only in the presence of "nice" twist angles like 0° or $\pm 90^\circ$ and becomes complicated in their absence, and often even when some of the α_i are "nice" angles.

Some Computational Considerations

In path control schemes, it is often necessary to solve for the inverse kinematic solutions of manipulator arms at a fairly high rate, sometimes as fast as 20-30 Hz.,[5] or faster. As such, computational efficiency is often an issue in manipulator inverse kinematic solution processes. Numerically iterative processes are unable to fulfill such requirements and are therefore not generally adopted, unless there does not exist a closed-form solution for the manipulator.

The structure of computation is also of importance. It is more efficient to generate all of the joint variables in parallel and to use lookup tables than to generate all of the angles serially. It is also much more efficient to generate only one solution than all solutions, specially when all of them are not required. Another time saving proceedure often adopted in practice is the generation of inverse kinematic solutions off-line, storage in a lookup table against a set of goal point positions, and then adjusting the solution to achieve the exact desired goal point position. The remaining orientation joint variables can then be computed by using the closed-form equations.

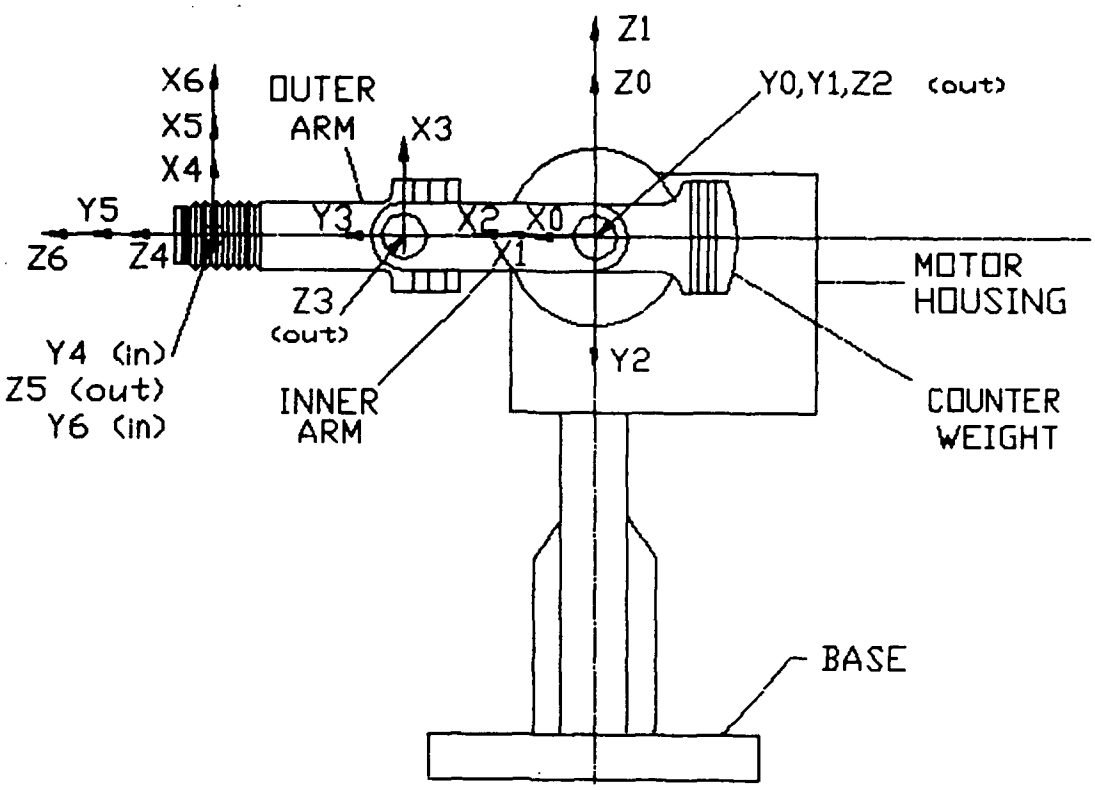


Figure 6. The Merlin 6500 Manipulator Frame Assignments - Side View.

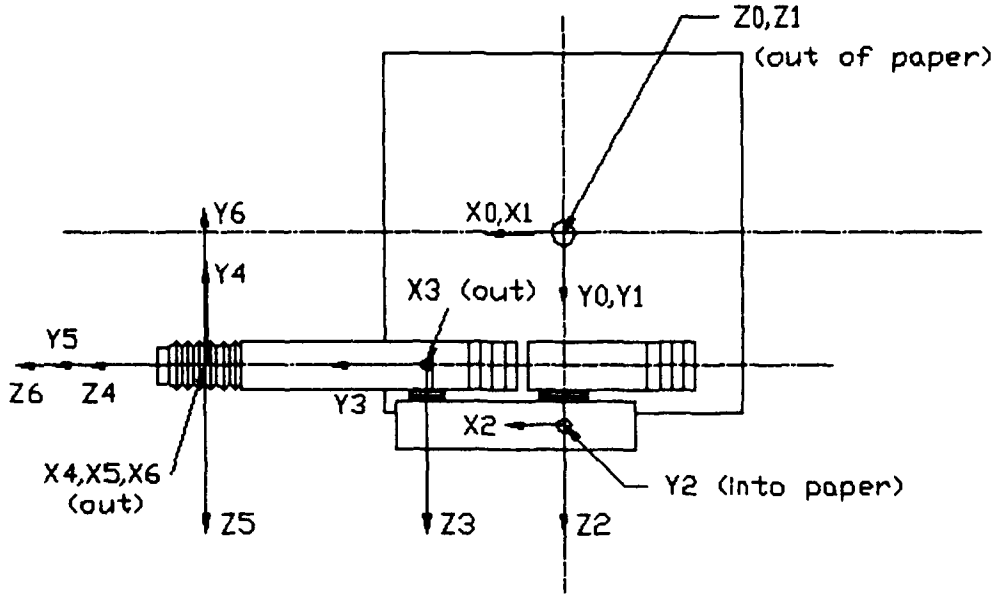


Figure 7. The Merlin 6500 Manipulator Frame Assignments - Top View.

V THE MERLIN 6500 KINEMATICS

Frame Assignments for the Merlin Manipulator

The first step involved in the kinematic analysis of manipulator mechanisms is the setup of Cartesian frames at each joint of the manipulator. This is done following the rules for frame assignment outlined in Chapter V, and is demonstrated in figures 6 and 7. Frame assignments have been performed in the 'Home' position, defined by the inner arm, the outer arm and the link between the wrist pin and the face plate being parallel to the floor and pointing towards the front of the robot.

The origin of the base frame {0} has been located at the intersection of the waist and shoulder axes, with the Z_0 axis aligned with the waist axis. This location of the origin of {0} offers the advantage of a similarity to anthropomorphous arm geometry.

The origin of frame {1} coincides with the origin of {0}, and {1} is coincident to {0} at the 'home' position. The origin of {2} is located at the center of the inner arm, with Z_2 positive from the origin of {2} in the direction formed from the waist to the shoulder.

Frame {3} has an origin located at the center of the outer arm. Z_3 lies along the axis of joint 3 and has a positive direction similar to Z_2, measured from the origin of {3}. Z_3 is always parallel to Z_2.

The origins of {4}, {5} and {6} are located at the center of the wrist pin. Z_4, Z_5 and Z_6 lie along their respective axes, with Z_4 and Z_6 positive towards the end of the arm and Z_5 positive coming out of the paper. The X_i and Y_i (i = 1 to 6) axes are set up according to the rules defined in Chapter 4.

The Kinematic Analysis Procedure

Following the assignment of frames at each joint, it is necessary to determine the kinematic parameters for the Merlin 6500 arm. These parameters are determined by using the rules outlined in Chapter 4. The direct kinematic analysis of the mechanism can then be performed by

forming the transformation matrices using (4.6) and concatenating them. There is always a unique result in the direct kinematic analysis of robotic arms.

A Summary of the Kinematic Parameters

Since all the joints are revolute, the joint variables are θ_i, (i = 1 to 6), where i denotes the joint number. The kinematic parameters are derived using the Denavit-Hartenburg convention, defined in Chapter V, and are summarized in Table 2.

Table 2. The Denavit-Hartenburg Parameters for the Merlin 6500 Left-Arm Manipulator

i	α_{i-1} (degrees)	a_{i-1} (inches)	d_2 (inches)	θ_i (degrees)	Kinematic Range (degrees)
1	0°	0"	0"	θ_1	$\pm 147^\circ$
2	-90°	0"	d_2 (18.915")	θ_2	$+ 56^\circ$ to $- 236^\circ$
3	0°	a_2(17.38")	d_3 (-6.915")	θ_3	$+ 56^\circ$ to $- 236^\circ$
4	-90°	0"	d_4 (17.24")	θ_4	$\pm 360^\circ$(continuous)
5	$+90^\circ$	0"	0"	θ_5	$\pm 90^\circ$
6	-90°	0"	0"	θ_6	$\pm 360^\circ$(continuous)

Note:

1) Right hand rule used (implying counterclockwise is + ve).

2) Source : Merlin System Operators Guide - Version 3.0 / June 1985.

The Direct Kinematic Solution for the Left-Arm Merlin

The general forward kinematic task is to compute the transformation matrix relating the tip of the end-effector to the global (or world)

coordinate frame of the robot. In the present case, we define the direct kinematic problem to be the computation of closed form equations that relate the position of the origin of {6}, and the orientation of the last link, with respect to {0}. It must be remembered here that the global frame of the robot is at a height of 46.45 inches above the base and that the hand roll frame, {6}, is located at the wrist pin.

The direct kinematic problem thus can be defined as the determination of the ${}^{0}_{6}T$ matrix, computable as

$$ {}^{0}_{6}T = {}^{0}_{1}T \cdot {}^{1}_{2}T \cdot {}^{2}_{3}T \cdot {}^{3}_{4}T \cdot {}^{4}_{5}T \cdot {}^{5}_{6}T \qquad (5.1) $$

The transformation matrices in (5.1) are given by:

$$ {}^{0}_{1}T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.2) $$

$$ {}^{1}_{2}T = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad d_2 \simeq 18.915" \qquad (5.3) $$

$$ {}^{2}_{3}T = \begin{bmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad a_2 \simeq 17.38" \qquad d_3 \simeq -6.915" \qquad (5.4) $$

$$
{}^{3}_{4}T = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ -s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad d_4 \simeq 17.24" \tag{5.5}
$$

$$
{}^{4}_{5}T = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.6}
$$

$$
{}^{5}_{6}T = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_6 & -c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.7}
$$

According to the principle of concatenation of transformations, developed in Chapter 3, we have

$$
{}^{4}_{6}T = {}^{4}_{5}T \cdot {}^{5}_{6}T \tag{5.8}
$$

Therefore,

$$
{}^{4}_{6}T = \begin{bmatrix} c_5c_6 & -c_5s_6 & -s_5 & 0 \\ s_6 & c_6 & 0 & 0 \\ s_5c_6 & -s_5s_6 & c_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.9}
$$

Further, using the principle of transformation matrix concatenation, we have

$$ {}^{3}_{6}T = {}^{3}_{4}T \cdot {}^{4}_{6}T \qquad (5.10) $$

i.e.

$$ {}^{3}_{6}T = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & -c_4s_5 & 0 \\ s_5c_6 & -s_5s_6 & c_5 & d_4 \\ -s_4c_5c_6 - c_4s_6 & s_4c_5s_6 - c_4c_6 & s_4s_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.11) $$

Now, since the joint axes for {2} & {3} are always parallel, we obtain ${}^{1}_{3}T$ using the trigonometric sum of angle formulas

$$ c_{23} = c_2c_3 - s_2s_3 \qquad \text{and} $$
$$ s_{23} = s_2c_3 + c_2s_3 $$

to yield a simple result.

Since

$$ {}^{1}_{3}T = {}^{1}_{2}T \cdot {}^{2}_{3}T \qquad (5.12) $$

we have

$$ {}^{1}_{3}T = \begin{bmatrix} c_{23} & -s_{23} & 0 & a_2c_2 \\ 0 & 0 & 1 & d_2+d_3 \\ -s_{23} & -c_{23} & 0 & -a_2s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.13) $$

Now, as

$$ {}^{1}_{6}T = {}^{1}_{3}T \cdot {}^{3}_{6}T \qquad (5.14) $$

we get

$$ {}^{1}_{6}T = \begin{bmatrix} {}^{1}r_{11} & {}^{1}r_{12} & {}^{1}r_{13} & {}^{1}p_{x} \\ {}^{1}r_{21} & {}^{1}r_{22} & {}^{1}r_{23} & {}^{1}p_{y} \\ {}^{1}r_{31} & {}^{1}r_{32} & {}^{1}r_{33} & {}^{1}p_{z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.15) $$

where

$$ {}^{1}r_{11} = c_{23}[c_4c_5c_6 - s_4s_6] - s_{23}s_5c_6 $$
$$ {}^{1}r_{12} = -c_{23}[c_4c_5s_6 + s_4c_6] + s_{23}s_5s_6 $$
$$ {}^{1}r_{13} = -[c_{23}c_4s_5 + s_{23}c_5] $$

$$ {}^{1}r_{21} = -[s_4c_5c_6 + c_4s_6] $$
$$ {}^{1}r_{22} = s_4c_5s_6 - c_4c_6 $$
$$ {}^{1}r_{23} = s_4s_5 $$

$$ {}^{1}r_{31} = -s_{23}[c_4c_5c_6 - s_4s_6] - c_{23}s_5c_6 $$
$$ {}^{1}r_{32} = s_{23}[c_4c_5s_6 + s_4c_6] + c_{23}s_5s_6 $$
$$ {}^{1}r_{33} = s_{23}c_4s_5 - c_{23}c_5 $$

$$ {}^{1}p_{x} = -d_4s_{23} + a_2c_2 $$
$$ {}^{1}p_{y} = d_2 + d_3 $$
$$ {}^{1}p_{z} = -d_4c_{23} - a_2s_2 $$

The final product of all six link transformations is given by

$$ {}^{0}_{6}T = {}^{0}_{1}T \cdot {}^{1}_{6}T \qquad (5.16) $$

which results in the final ${}^{0}_{6}T$ matrix, given by,

$$
{}^{0}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.17)
$$

where

$$
\begin{aligned}
r_{11} &= c_1[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6] + s_1[s_4c_5c_6 + c_4s_6] \\
r_{21} &= s_1[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6] - c_1[s_4c_5c_6 + c_4s_6] \\
r_{31} &= -s_{23}[c_4c_5c_6 - s_4s_6] - c_{23}s_5c_6
\end{aligned}
$$

$$
\begin{aligned}
r_{12} &= c_1[-c_{23}(c_4c_5s_6 + s_4c_6) + s_{23}s_5s_6] - s_1[s_4c_5s_6 - c_4c_6] \\
r_{22} &= s_1[-c_{23}(c_4c_5s_6 + s_4c_6) + s_{23}s_5s_6] + c_1[s_4c_5s_6 - c_4c_6] \\
r_{32} &= s_{23}[c_4c_5s_6 + s_4c_6] + c_{23}s_5s_6
\end{aligned}
$$

$$
\begin{aligned}
r_{13} &= -c_1[c_{23}c_4s_5 + s_{23}c_5] - s_1[s_4s_5] \\
r_{23} &= -s_1[c_{23}c_4s_5 + s_{23}c_5] + c_1[s_4s_5] \\
r_{33} &= s_{23}c_4s_5 - c_{23}c_5
\end{aligned}
$$

$$
\begin{aligned}
p_x &= c_1[-d_4s_{23} + a_2c_2] - s_1(d_2 + d_3) \\
p_y &= s_1[-d_4s_{23} + a_2c_2] + c_1(d_2 + d_3) \\
p_z &= -d_4c_{23} - a_2s_2
\end{aligned}
$$

The transformation matrix ${}^{0}_{6}T$, given by (5.17), completely defines and locates the position of the wrist pin and orientation of the link connecting the tip of the Merlin arm to the wrist pin, with respect to the base frame. The position of the tip of the Merlin manipulator with respect to the base frame is easily computable from the above transformation matrix. This requires the addition of the product of each term in the 'approach' vector (the third column vector of the ${}^{0}_{6}T$

matrix) and the distance between the tip of the arm and the wrist pin (or the distance between the tip of the arm and the point under consideration), to the corresponding term in the position vector (fourth column in the ${}^{0}_{6}T$ matrix). Thus, if 'd_6' defines the distance between the tip of the Merlin 6500 arm and the wrist pin (or the point under consideration), then the ${}^{0}_{6}T$ matrix can be modified so that the position data provided by the transformation matrix ${}^{0}_{6}T$ refers to the end of the arm, as follows :

$$\begin{aligned} p'_x &= {}^{0}_{6}T(1,4) = p_x + d_6 \cdot {}^{0}_{6}T(1,3) \\ p'_y &= {}^{0}_{6}T(2,4) = p_y + d_6 \cdot {}^{0}_{6}T(2,3) \\ p'_z &= {}^{0}_{6}T(3,4) = p_z + d_6 \cdot {}^{0}_{6}T(3,3) \end{aligned} \qquad (5.18)$$

The direct kinematic solution could have alternatively been performed by setting the origin of {6} at the tip of the Merlin arm (or at the point under consideration), instead of the wrist pin. This method would, however, cause computational complications when performing the inverse kinematic solution for the arm, since the solution process by Piepers method requires three axes intersecting at a point and the origin of the three frames for these axes are set at the point of intersection.

Direct Kinematics for the Right-arm Merlin

We now need to develop the direct kinematics for the right shouldered Merlin manipulator. This can be performed either by repeating the above process completely for the right arm Merlin manipulator, or by utilizing the solution developed for the left arm manipulator with adjustments being made to the values of the kinematic parameters. The former process involves re-assigning frames, determining the Denavit-Hartenburg parameters and then computing ${}^{0}_{6}T$ for the right arm manipulator. The latter process maintains the frame assignments made for the left arm while adjusting the numeric values of those parameters that would be affected by the conversion of the left

arm to a right arm manipulator, and using the direct kinematic equations for the left arm robot, given by (5.17).

A close examination of the left- and right- shouldered arms reveals that they differ kinematically at the shoulder only. We thus adjust the Denavit-Hartenburg parameters indicated by d_2 and d_3 to be $d_2 \simeq -18.915$ inches and $d_3 \simeq 6.915$ inches. Equations (5.17), when solved for with the above values of d_2 and d_3, result in the direct kinematic solution for the right arm Merlin manipulator.

The Inverse Kinematic Solution for the Left-Arm Merlin.

Since the last three axes of the Merlin manipulator intersect at the wrist pin, we adopt an algebraic method (Pieper's) to solve for the inverse kinematic solution. Since we may be given the position & orientation of the hand-roll plate, and the origins of frames {4}, {5} and {6} are located at the wrist pin, we need to account for the distance between the wrist-pin and the tip of the hand roll plate, which is $\simeq$ 3.5". This affects the position vector only - the orientation vector remains unchanged.

The transformation matrix defining the position and orientation is given by (5.17), and is

$$ {}^{0}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

where r_{11}, r_{12}, ... r_{33}, p_x, p_y, p_z are specified by the kinematic equations given in (5.17).

Let d_6 be the distance measured from the tool mounting surface to the wrist pin ($d_6 \simeq 3.5$") and the position of the tool mounting surface be given by a vector $p' = \{p'_x\ p'_y\ p'_z\}^T$, where

$$p' = \begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} + d_6 \cdot \begin{bmatrix} r_{13} \\ r_{23} \\ r_{33} \end{bmatrix} \qquad (5.19)$$

Therefore, the position of the wrist pin is specified by

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} - d_6 \begin{bmatrix} -c_1[c_{23}c_4s_5 + s_{23}c_5] - s_1[s_4s_5] \\ -s_1[c_{23}c_4s_5 + s_{23}c_5] + c_1[s_4s_5] \\ s_{23}c_4s_5 - c_{23}c_5 \end{bmatrix} \qquad (5.20)$$

We now examine the kinematic parameters to determine the number of solutions that will be obtained when solving the inverse kinematics of the Merlin robot.

Since $a_1 = a_3 = a_5 = 0$, we determine (from Chapter 4) that the number of solutions for the left shouldered Merlin arm will be four in number. Since a_1, a_3 and a_5 are unaffected by the shoulders configuration, four further solutions will be obtained for the right arm Merlin robot. This results in a total set of eight solutions for the inverse kinematics of the Merlin robot. These solutions can be seen to include the following configurations for each of the left and right arms :

1) Inner arm up, outer arm down, wrist roll, wrist pitch.
2) Inner arm down, outer arm up, wrist roll, wrist pitch.
3) Inner arm up, outer arm down, wrist 'flipped' over.
4) Inner arm down, outer arm up, wrist 'flipped' over.

Four similar solutions exist for the Merlin right-shouldered manipulator.

We now proceed to solve the inverse kinematics of the Merlin left arm manipulator. The inverse kinematic solution process requires solving

$$ {}^{0}_{6}T = {}^{0}_{1}T(\theta_1) \cdot {}^{1}_{2}T(\theta_2) \cdot {}^{2}_{3}T(\theta_3) \cdot {}^{3}_{4}T(\theta_4) \cdot {}^{4}_{5}T(\theta_5) \cdot {}^{5}_{6}T(\theta_6) \qquad (5.21) $$

for θ_i, i = 1 to 6, when ${}^{0}_{6}T$ is given as numeric values, with the position vector of ${}^{0}_{6}T$ having been adjusted according to (5.20), if necessary.

Putting the dependence of θ_1 on the left side of the equation gives

$$ \left[{}^{0}_{1}T(\theta_1)\right]^{-1} \cdot {}^{0}_{6}T = {}^{1}_{2}T(\theta_2) \cdot {}^{2}_{3}T(\theta_3) \cdot {}^{3}_{4}T(\theta_4) \cdot {}^{4}_{5}T(\theta_5) \cdot {}^{5}_{6}T(\theta_6) \qquad (5.22) $$

Inverting ${}^{0}_{1}T$, we rewrite (5.22) as

$$ \begin{bmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^{1}_{6}T \qquad (5.23) $$

where ${}^{1}_{6}T$ is given by (5.15).

Equating the (2,4) elements from both sides of (5.23), we get

$$ -s_1 p_x + c_1 p_y = d_2 + d_3 \qquad (5.24) $$

Substituting

$$ \left.\begin{aligned} p_x &= \rho \, \mathrm{Cos}\, \phi \qquad \text{and} \\ p_y &= \rho \, \mathrm{Sin}\, \phi \end{aligned}\right] \qquad (5.25) $$

$$ \left.\begin{aligned} \text{where} \quad \rho &= \sqrt{p_x^{\,2} + p_y^{\,2}} \\ \text{and} \quad \phi &= \mathrm{Atan2}\,(p_y, p_x) \end{aligned}\right] \qquad (5.26) $$

into equation (5.24), we get

$$\rho(\text{Sin}\phi\ \text{Cos}\theta_1 - \text{Cos}\phi\ \text{Sin}\theta_1) = d_2 + d_3$$

which results in

$$\text{Sin}(\phi - \theta_1) = \frac{d_2 + d_3}{\rho} \tag{5.27}$$

Using

$$\text{Sin}^2 A + \text{Cos}^2 A = 1,$$

we get

$$\text{Cos}(\phi - \theta_1) = \pm \sqrt{1 - \left[\frac{d_2 + d_3}{\rho}\right]^2} \tag{5.28}$$

Since we know $\text{Sin}(\phi - \theta_1)$ and $\text{Cos}(\phi - \theta_1)$, we find $(\phi - \theta_1)$ as

$$(\phi - \theta_1) = \text{Atan2}\left[\left[\frac{d_2 + d_3}{\rho}\right], \pm \sqrt{1 - \left[\frac{d_2 + d_3}{\rho}\right]^2}\right]$$

Using the value of ρ from (5.26), we get

$$(\phi - \theta_1) = \text{Atan2}\left[\left[d_2 + d_3\right], \pm \sqrt{p_x^2 + p_y^2 - \left[d_2 + d_3\right]^2}\right] \tag{5.29}$$

i.e.

$$\theta_1 = \text{Atan2}(p_y, p_x) - \text{Atan2}\left[\left[d_2 + d_3\right], \pm \sqrt{p_x^2 + p_y^2 - \left[d_2 + d_3\right]^2}\right] \tag{5.30}$$

In equation (5.30), we have utilized the Atan2 function to determine the value of θ_1. Use of the cosine or arc sine function would lead to inaccurate, inconsistent and ill-conditioned solutions, since

the accuracy of the arc cosine function in determining the angle is dependant on the angle [Cos θ = Cos(- θ)], while, when Sin θ approaches zero, $\theta = 0^\circ$ or $\pm 180^\circ$. A more consistent approach is to use the Atan2 function, which returns the value of θ adjusted to the proper quadrant. The Atan2 function is defined as follows

$$\theta = \text{Atan2}(y, x) = \begin{cases} 0^\circ \le \theta \le 90^\circ & \text{for} \quad +x \text{ and } +y \\ 90^\circ \le \theta \le 180^\circ & \text{for} \quad -x \text{ and } +y \\ -180^\circ \le \theta \le -90^\circ & \text{for} \quad -x \text{ and } -y \\ -90^\circ \le \theta \le 0^\circ & \text{for} \quad +x \text{ and } -y \end{cases}$$

Note that, in (5.30), there are two possible solutions to θ_1, depending on the $\pm$ sign in the second term of the equation. The positive solution is obtained for the left arm Merlin 6500 manipulator, while the negative solution represents the inverse kinematic solution for θ_1 for the right arm Merlin 6500 manipulator with different frame assignments than those made for the left arm.

Since θ_1 is now known, we now know the left side of equation (5.22) and (5.23).

Equating the (1,4) elements of (5.23), we have

$$c_1 p_x + s_1 p_y = -d_4 s_{23} + a_2 c_2 \qquad (5.31)$$

Equating the (3,4) elements of equation (5.23), we have

$$p_z = -d_4 c_{23} - a_2 s_2 \qquad (5.32)$$

Squaring equations (5.24), (5.31) and (5.32) and adding, we get

$$p_x^2 + p_y^2 + p_z^2 = a_2^2 + d_4^2 - 2a_2 d_4 s_3 + \left[d_2 + d_3\right]^2$$

Therefore,

$$-2a_2d_4s_3 = p_x^2 + p_y^2 + p_z^2 - a_2^2 - \left[d_2 + d_3\right]^2 - d_4^2$$

which results in

$$s_3 = \frac{-1}{2a_2d_4}\left[p_x^2 + p_y^2 + p_z^2 - a_2^2 - \left[d_2 + d_3\right]^2 - d_4^2\right] \qquad (5.33)$$

Since

$$s_3^2 + c_3^2 = 1$$

we have

$$c_3 = \pm\sqrt{1 - s_3^2}$$

Therefore

$$\theta_3 = \text{Atan2}\left[\, s_3, \pm\sqrt{1 - s_3^2}\,\right] \qquad (5.34)$$

where

$$s_3 = \frac{-1}{2a_2d_4}\left[\, p_x^2 + p_y^2 + p_z^2 - a_2^2 - \left[d_2 + d_3\right]^2 - d_4^2 \,\right] \qquad (5.35)$$

Thus, θ_3 can have two values, depending on the ± sign used in (5.34). Each of the solutions represents the elbow up or down solution. Both of the above solutions for θ_3 are valid for the left and right arm Merlin robot. The values of d_2 and d_3 that are used in (5.35) will

depend on whether the arm solution desired is for the left or for the right arm.

Equation (5.22) can now be written so that we have the left side as a function of the known variables θ_1 and θ_3 and the unknown θ_2, as

$$\left[{}^{0}_{3}T(\theta_2)\right]^{-1} \cdot {}^{0}_{6}T = {}^{3}_{4}T(\theta_4) \cdot {}^{4}_{5}T(\theta_5) \cdot {}^{5}_{6}T(\theta_6) \qquad (5.36)$$

Since

$${}^{0}_{3}T = {}^{0}_{1}T \cdot {}^{1}_{3}T$$

we have

$${}^{0}_{3}T(\theta_2) = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & -s_1 & a_2c_1c_2 - s_1(d_2+d_3) \\ s_1c_{23} & -s_1s_{23} & c_1 & a_2s_1c_2 + c_1(d_2+d_3) \\ -s_{23} & -c_{23} & 0 & -a_2s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.37)$$

We invert $\left[{}^{0}_{3}T(\theta_2)\right]$ using (3.18), to get

$$\left[{}^{0}_{3}T\right]^{-1} = \begin{bmatrix} c_1c_{23} & s_1c_{23} & -s_{23} & -a_2c_3 \\ -c_1s_{23} & -s_1s_{23} & -c_{23} & a_2s_3 \\ -s_1 & c_1 & 0 & -(d_2+d_3) \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.38)$$

Equation (5.36) can now be written as

$$\begin{bmatrix} c_1c_{23} & s_1c_{23} & -s_{23} & -a_2c_3 \\ -c_1s_{23} & -s_1s_{23} & -c_{23} & a_2s_3 \\ -s_1 & c_1 & 0 & -(d_2+d_3) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^3_6T \qquad (5.39)$$

where

$${}^3_6T = \begin{bmatrix} c_4c_5c_6-s_4s_6 & -c_4c_5s_6-s_4c_6 & -c_4s_5 & 0 \\ s_5c_6 & -s_5s_6 & c_5 & d_4 \\ -s_4c_5c_6-c_4s_6 & s_4c_5s_6-c_4c_6 & s_4s_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equating the (1,4) and (2,4) elements of equation (5.39), we have

$$\left.\begin{aligned} c_1c_{23}p_x + s_1c_{23}p_y - s_{23}p_z - a_2c_3 &= 0 \\ -c_1s_{23}p_x - s_1s_{23}p_y - c_{23}p_z + a_2s_3 &= d_4 \end{aligned}\right] \qquad (5.40)$$

Taking the known terms to the right side of the equation, we have

$$\begin{aligned} c_{23}(c_1p_x + s_1p_y) - s_{23}p_z &= a_2c_3 \\ s_{23}(c_1p_x + s_1p_y) + c_{23}p_z &= a_2s_3 - d_4 \end{aligned} \qquad (5.41)$$

Let

$$\begin{aligned} A &= c_1p_x + s_1p_y \\ B &= p_z \\ C &= a_2c_3 \\ D &= a_2s_3 - d_4 \end{aligned}$$

Therefore, (5.41) now becomes

$$\left.\begin{aligned} c_{23}A - s_{23}B &= C \\ c_{23}B + s_{23}A &= D \end{aligned}\right] \qquad (5.42)$$

Solving for c_{23} and s_{23} by Cramers rule, we have

$$c_{23} = \frac{(a_2s_3 - d_4)p_z + a_2c_3(c_1p_x + s_1p_y)}{p_z^2 + (c_1p_x + s_1p_y)^2} \quad , \qquad (5.43)$$

$$s_{23} = \frac{(a_2s_3 - d_4)(c_1p_x + s_1p_y) - a_2c_3p_z}{p_z^2 + (c_1p_x + s_1p_y)^2} \quad , \qquad (5.44)$$

and

$$\theta_{23} = \text{Atan2}\Big[\{(a_2s_3 - d_4)(c_1p_x + s_1p_y) - a_2c_3p_z\}, \{(a_2s_3 - d_4)p_z + a_2c_3(c_1p_x + s_1p_y)\Big] \qquad (5.45)$$

Due to the four possible combination of solutions of θ_1 and θ_3, there will be a total of four possible solutions for θ_{23}. As such, the four possible solutions for θ_2 are computed as

$$\theta_2 = \theta_{23} - \theta_3 \qquad (5.46)$$

where the appropriate solution for θ_3 is used when forming the difference. Since the computed value of θ_1 is used in solving (5.45), and hence (5.46), the left arm (positive) solution for θ_1 provides the left arm solution for θ_2 (which account for two of the four solutions obtained above), while the right arm solution for θ_1 (the negative solution) provides that solution for θ_2 which is valid for the right arm only (and which account for the remaining two solutions obtained

for θ_2).

We now know the entire left side of equation (5.39). Equating the (1,3) and (3,3) elements from both sides of (5.39), we get

$$\left.\begin{aligned} r_{13}c_1c_{23} + r_{23}s_1c_{23} - r_{33}s_{23} &= -c_4s_5 \\ -r_{13}s_1 + r_{23}c_1 &= s_4s_5 \end{aligned}\right] \qquad (5.47)$$

If, in (5.47), $s_5 \neq 0$, we solve for θ_4 as follows

$$s_4 = (-r_{13}s_1 + r_{23}c_1)\ \frac{1}{s_5}$$

and

$$c_4 = (-r_{13}c_1c_{23} - r_{23}s_1c_{23} + r_{33}s_{23})\ \frac{1}{s_5}$$

Therefore

$$\theta_4 = \text{Atan2}\left[(-r_{13}s_1 + r_{23}c_1),\ (-r_{13}c_1c_{23} - r_{23}s_1c_{23} + r_{33}s_{23})\right] \qquad (5.48)$$

If, however, $s_5 = 0$, then $\theta_5 = 0^\circ$ or 180°, and the manipulator is in a singular configuration, in which the wrist roll and hand roll axes (z_4 and z_6) line up and cause the same motion of the last link of the robot. In such a case, all that can be solved for is the sum or difference of θ_4 and θ_6. This condition of singularity is detected by checking to see if the two arguments of the Atan2 function of equation (5.48) are close to zero. If they are, θ_4 should be chosen to be the present value (or any other arbitrary value) of the wrist roll angle. When θ_6 is computed at the last stage using the present (or arbitrary) value of θ_4, it is adjusted according to the value chosen for θ_4.

Considering equation (5.21) again, we now know θ_1, θ_2, θ_3 and θ_4. So we rewrite equation (5.21) to get all the knowns on the left side, as follows:

$$\left[{}^{0}_{4}T\right]^{-1} \cdot {}^{0}_{6}T = {}^{4}_{5}T(\theta_5) \cdot {}^{5}_{6}T(\theta_6) \tag{5.49}$$

We know that

$${}^{0}_{4}T = {}^{0}_{3}T \cdot {}^{3}_{4}T$$

Since

$$(AB)^{-1} = (B)^{-1} \cdot (A)^{-1}$$

we have

$$\left[{}^{0}_{4}T\right]^{-1} = \left[{}^{3}_{4}T\right]^{-1} \cdot \left[{}^{0}_{3}T\right]^{-1}$$

We compute $\left[{}^{3}_{4}T\right]^{-1}$ using (3.18) and (5.5) as

$$\left[{}^{3}_{4}T\right]^{-1} = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ -s_4 & 0 & -c_4 & 0 \\ 0 & 1 & 0 & -d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.50}$$

Having computed $\left[{}^{0}_{3}T\right]^{-1}$ in (5.38), we solve

$$\left[{}^{0}_{4}T\right]^{-1} = \left[{}^{3}_{4}T\right]^{-1} \cdot \left[{}^{0}_{3}T\right]^{-1}$$

to get

$$\begin{bmatrix}{}^{0}_{4}T\end{bmatrix}^{-1} = \begin{bmatrix} c_1c_{23}c_4+s_1s_4 & s_1c_{23}c_4-c_1s_4 & -s_{23}c_4 & -a_2c_3c_4+s_4(d_2+d_3) \\ -c_1c_{23}s_4+s_1c_4 & -s_1c_{23}s_4-c_1c_4 & s_{23}s_4 & a_2c_3s_4+c_4(d_2+d_3) \\ -c_1s_{23} & -s_1s_{23} & -c_{23} & a_2s_3-d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.51}$$

Equation (5.49) can now be written as

$$\begin{bmatrix}{}^{0}_{4}T\end{bmatrix}^{-1} \cdot {}^{0}_{6}T = {}^{4}_{6}T \tag{5.52}$$

where ${}^{0}_{6}T$ is given by (5.17) and ${}^{4}_{6}T$ by (5.9)

Equating the (1, 3) and (3, 3) terms of (5.52), we have

$$\left.\begin{aligned} -r_{13}(c_1c_{23}c_4 + s_1s_4) - r_{23}(s_1c_{23}c_4 - c_1s_4) + r_{33}s_{23}c_4 = s_5 \\ -c_1s_{23}r_{13} - s_1s_{23}r_{23} - c_{23}r_{33} = c_5 \cdot \end{aligned}\right] \tag{5.53}$$

We therefore solve for θ_5 as

$$\theta_5 = \text{Atan2}(s_5, c_5) \tag{5.54}$$

where s_5 and c_5 are given by (5.53).

Rewriting (5.21) to get the known terms on the left side, we have

$$\begin{bmatrix}{}^{0}_{5}T\end{bmatrix}^{-1} \cdot {}^{0}_{6}T = {}^{5}_{6}T(\theta_6) \tag{5.55}$$

As before

$${}^{0}_{5}T = {}^{0}_{4}T \cdot {}^{4}_{5}T$$

Therefore

$$\begin{bmatrix}{}^{0}_{5}T\end{bmatrix}^{-1} = \begin{bmatrix}{}^{4}_{5}T\end{bmatrix}^{-1} \cdot \begin{bmatrix}{}^{0}_{4}T\end{bmatrix}^{-1} \tag{5.56}$$

We compute $\begin{bmatrix}{}^{4}_{5}T\end{bmatrix}^{-1}$ using (3.18) and (5.6) as

$$\begin{bmatrix}{}^{4}_{5}T\end{bmatrix}^{-1} = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ -s_5 & 0 & c_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Having computed $\begin{bmatrix}{}^{0}_{4}T\end{bmatrix}^{-1}$ in (5.51), we solve for $\begin{bmatrix}{}^{0}_{5}T\end{bmatrix}^{-1}$ as

$$\begin{bmatrix}{}^{0}_{5}T\end{bmatrix}^{-1} = \begin{bmatrix} c_5(c_1c_{23}c_4+s_1s_4)-c_1s_{23}s_5 & c_5(s_1c_{23}c_4-c_1s_4)-s_1s_{23}s_5 \\ -s_5(c_1c_{23}c_4+s_1s_4)-c_1s_{23}c_5 & -s_5(s_1c_{23}c_4-c_1s_4)-s_1s_{23}c_5 \\ c_1c_{23}s_4 - s_1c_4 & s_1c_{23}s_4 + c_1c_4 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -s_{23}c_4c_5-c_{23}s_5 & \{c_5(-a_2c_3c_4+s_4(d_2+d_3)+s_5(a_2s_3-d_4)\} \\ s_{23}c_4s_5-c_{23}c_5 & \{-s_5(-a_2c_3c_4+s_4(d_2+d_3)+c_5(a_2s_3-d_4)\} \\ -s_{23}s_4 & -a_2c_3s_4 - c_4(d_2+d_3) \\ 0 & 1 \end{bmatrix} \tag{5.57}$$

Since we know $\begin{bmatrix}{}^{5}_{6}T\end{bmatrix}$ from (5.7), we equate the (1,1) and the (3,1) elements of (5.55), to get

$$\begin{bmatrix} \{c_5(c_1c_{23}c_4 + s_1s_4) - c_1s_{23}s_5\}r_{11} + \\ \{c_5(s_1c_{23}c_4 - c_1s_4) - s_1s_{23}s_5\}r_{21} - \{s_{23}c_4c_5 + c_{23}s_5\}r_{31} = c_6 \\ - \{c_1c_{23}s_4 - s_1c_4\}r_{11} - \{s_1c_{23}s_4 + c_1c_4\}r_{21} + \{s_{23}s_4\}r_{31} = s_6 \end{bmatrix} \tag{5.58}$$

We therefore determine θ_6 by

$$\theta_6 = \text{Atan2}(s_6, c_6) \tag{5.59}$$

where s_6 and c_6 are given by (5.58).

Since there are two possible solutions for each of θ_1 and θ_3, equation (5.59) results in a total of four solutions for θ_6. If the positive value for θ_1 is used in solving (5.59), then the solution obtained for θ_6 by (5.59) is for the left arm Merlin robot, while if the negative solution for θ_1 is used, we obtain the solution for the right arm Merlin robot.

Additional solutions are obtained by flipping over the wrist of the manipulator, and are given by

$$\left.\begin{aligned} \theta_4' &= \theta_4 + 180^1 , \\ \theta_5' &= -\theta_5 \qquad \text{and} \\ \theta_6' &= \theta_6 + 180^1 . \end{aligned}\right] \tag{5.60}$$

After all of the above eight solutions have been computed, some (or all) of them may have to be discarded because of joint limit violations. Of the remaining valid solutions, it is advisable in most cases to choose the one closest to the current configuration of the manipulator.

Inverse Kinematics for the Right-arm Merlin Manipulator

Of the above eight solutions which constitute the solution set for the inverse kinematics for the left and right arm Merlin 6500 manipulator, those solutions obtained using the positive solution for θ_1 represent the inverse kinematic solution for the left arm Merlin robot, while the four solutions obtained using the negative solution for θ_1 represent the solution set for the inverse kinematics for the right arm

Merlin robot with adjusted frames.

An alternative procedure for developing the closed-form inverse kinematics for the right arm Merlin manipulator involves using the same set of equations developed for the left arm Merlin but adjusting the kinematic parameters involved in converting from a left arm Merlin robot into a right arm Merlin. As explained in Chapter 5, this essentially involves retaining the orientation of the frames assigned to the individual joints and negating those parameters that will effect the conversion of the left arm to a right arm Merlin robot, viz. d_2 and d_3. Thus, setting $d_2 \simeq -18.915"$ and $d_3 \simeq 6.915"$ will result in a left arm Merlin becoming a right arm Merlin robot. Using the above adjusted parameter values for the right arm parameters in the left arm (θ_1 positive solution set) equations results in the solution being obtained for the right arm Merlin 6500 manipulator. The major advantage of this process is the avoidance of new frame assignments, reduced code in computer implementations as well as the fact that new frame assignments and joint angle measurement processes do not have to be followed. This process has therefore been adopted in the computer implementation of the equations (Appendix IV).

VI WORKSPACE DEVELOPMENT FOR THE MERLIN 6500 ROBOT ARM

Workspaces of Manipulators

The working volume of a manipulator is called the manipulators workspace. The workspace of any manipulator is defined as the set of positions that the end-effector can achieve when the joints vary over the full range of possible values[5].

Manipulator workspaces are divided into reachable and dexterous workspaces. The reachable workspace for a manipulator arm is defined as that volume of space that the manipulator's end-effector can reach in at least one orientation (Appendix A1.10). The dexterous workspace of a manipulator is defined as that volume of space which the end-effector can reach with all possible orientations (Appendix A1.11). The dexterous workspace of a manipulator is always a sub-set of the reachable workspace.

Each of the dexterous and reachable workspaces of a manipulator possess an outer and inner boundary. The outer boundaries are determinable by the set of positions in Cartesian space corresponding to the tip of the end-effector (or tool) of the manipulator, when the joints of the manipulator are taken through their full range of motion. The inner boundary for manipulators (with the last three axes intersecting at a common point) are defined by the set of positions in Cartesian space that the point of intersection goes through when the joint motions are taken through their full range of motion.

Previous work on manipulator workspace generation has been performed by a variety of methods[9,10,11]. Since manipulator workspaces are geometrically complex, it has been found easier to obtain an understanding of their shape by developing their workspaces in two dimensions i.e. in planes. The complete workspace is a composite of the two dimensional workspaces in all three dimensions and can be formed by overlaying the two perpendicularly, and aligning the axes common to both planes. The workspaces that are commonly developed for manipulators consist of the horizontal workspace (a projection of the manipulator workspace onto a horizontal plane) and the vertical workspace (a projection of the workspace onto a vertical plane).

The process of two dimensional workspace development consists of

the division of the degrees-of-freedom of the manipulator into those that act in the vertical plane, those that act in a horizontal plane and those successive degrees-of-freedom that can be combined together to produce a motion with an axis perpendicular to the axes in which each joint acts separately.

The Horizontal Workspace of the Merlin 6500 manipulator

The only possible link motions in the horizontal plane that the Merlin 6500 arm possesses consist of the waist rotation over a range of 294 degrees and the wrist yaw over a range of 180 degrees. The wrist yawing motion is formed by a wrist roll of 90 degrees, followed by a wrist pitching motion over the full range of 180 degrees. The manipulator is therefore kinematically representable as shown in figure 8.

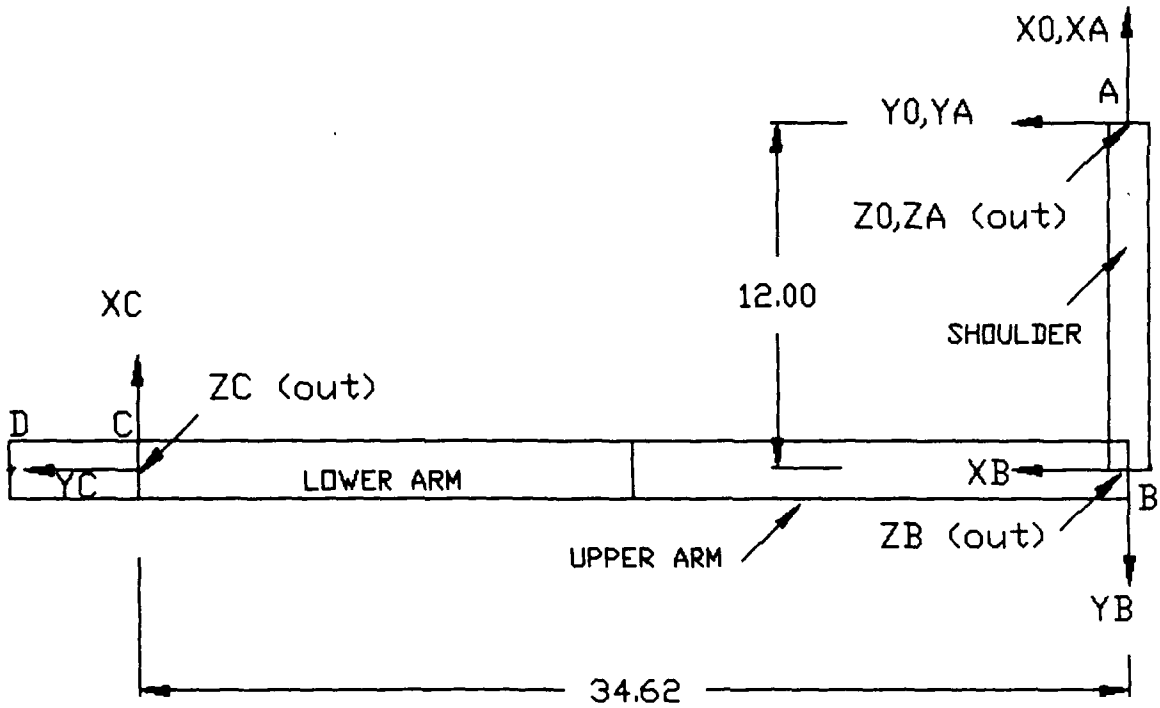


Figure 8. The Horizontal Plane Representation of the Merlin 6500 arm

The Merlin 6500 horizontal degrees-of-freedom can be assumed to form a manipulator with the kinematic parameters as in Table 3.

Table 3. The Merlin 6500 Horizontal Plane Representation and the Corresponding Denavit-Hartenburg Parameters.

i	α_{i-1} °	a_{i-1} "	d_i "	θ_i° and range
A	0°	0	0	$\pm 147^\circ$
B	0°	-d ($\simeq$ 12")	0	90° constant
C	0°	L" ($\simeq$ 34.62")	0"	0 to -180°

Using these kinematic parameters, the transformation matrices that relate each frame with respect to the previous frame can be computed using the general form of the ${}^{i-1}_{i}T$ matrix, given in equation (4.6). It is required here to determine both the end-points of each link. These end-points can be determined from the last column, i.e. from the position vector, of the concatenated transformation matrices. Since the intersection of the waist and shoulder axes is always at a fixed point, the graphical origin is located at this point. The process of transform concatenation can be performed so that the determination of the end-points is made during the concatenation process.

Using the general form of the transformation matrix developed in (4.6), we develop the kinematic transform matrices relating each link-joint system to its previous one. Thus,

$$ {}^{0}_{A}T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6.1) $$

$$
{}^{A}_{B}T = \begin{bmatrix} c_2 & -s_2 & 0 & -d \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.2}
$$

$$
{}^{B}_{C}T = \begin{bmatrix} c_3 & -s_3 & 0 & L \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.3}
$$

Since

$$
{}^{0}_{B}T = {}^{0}_{A}T \cdot {}^{A}_{B}T
$$

we have,

$$
{}^{0}_{B}T = \begin{bmatrix} c_{12} & -s_{12} & 0 & -dc_1 \\ s_{12} & c_{12} & 0 & -ds_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.4}
$$

Further, since

$$
{}^{0}_{C}T = {}^{0}_{B}T \cdot {}^{B}_{C}T \tag{6.5}
$$

we have

$$
{}^{0}_{C}T = \begin{bmatrix} c_{123} & -s_{123} & 0 & (-dc_1 + Lc_{12}) \\ s_{123} & c_{123} & 0 & (-ds_1 + Ls_{12}) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6.6)
$$

From the position vector (last column) of equations (6.4) and (6.6), we can extract the graphical co-ordinates of the end-points of the lines representing the kinematic skeleton of the Merlin 6500 manipulator arm.

We finally need to compute the position of D with respect to the origin A. From figure 8, we note that {D} is located on the sliding vector (Y-axis) of {C}, at a distance d_6" from {C}. The position of {D} in each of the X- and Y- coordinate directions is given by

$$
\left.\begin{aligned} D_x &= C_x + d_6 \cdot (-s_{123}) \\ D_y &= C_y + d_6 \cdot (c_{123}) \end{aligned}\right] \qquad (6.7)
$$

The graphical coordinates of each point A, B and C are therefore determinable from (6.4), (6.6) and (6.7), and are

$$
\left.\begin{aligned} A_x &= 0 \; ; & A_y &= 0 \\ B_x &= -dc_1 \; ; & B_y &= -ds_1 \\ C_x &= -dc_1 + Lc_{12} \; ; & C_y &= -ds_1 + Ls_{12} \\ D_x &= -dc_1 + Lc_{12} - d_6 s_{123} \; ; & D_y &= -ds_1 + Ls_{12} + d_6 c_{123} \end{aligned}\right] \qquad (6.8)
$$

where 's' denotes the sine, and 'c' the cosine, of the sum of the joint angles in the subscript.

The Vertical Workspace of the Merlin 6500 Manipulator

We now proceed to develop the vertical plane workspace of the Merlin 6500 left arm manipulator.

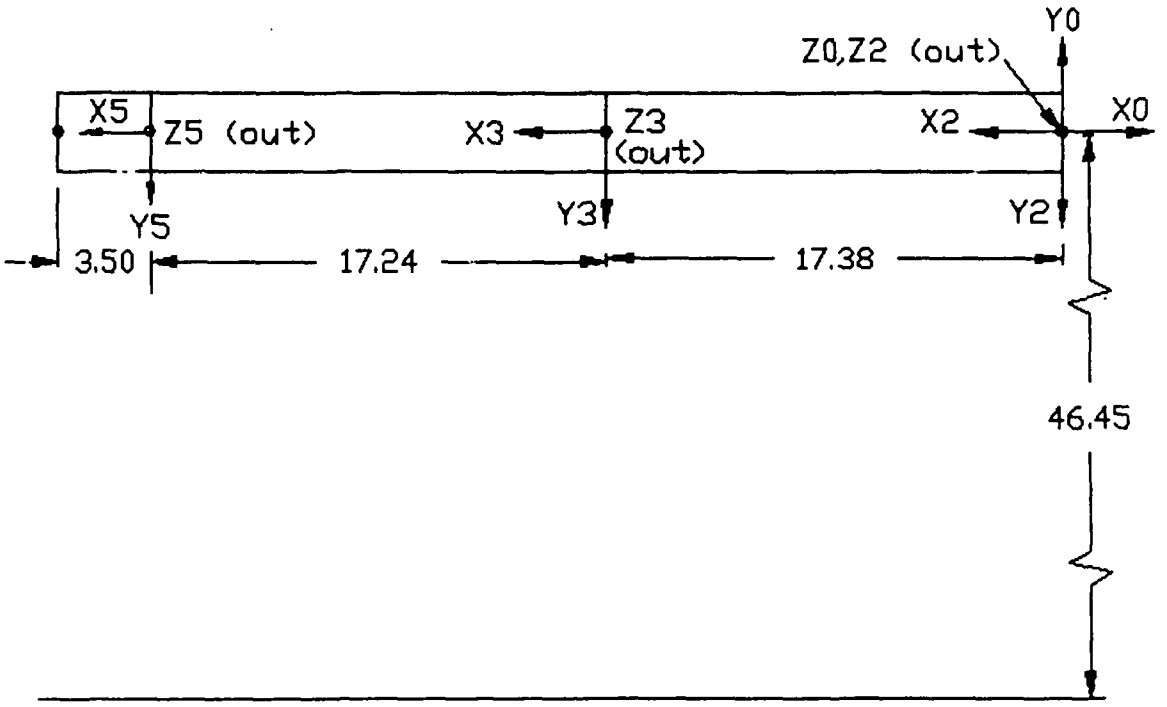


Figure 9. The Vertical Plane Representation of the Merlin 6500 arm

In the vertical plane, the Merlin 6500 arm can be represented by its degrees-of-freedom which allow motion only in that plane. These degrees-of-freedom are the Shoulder Pitch, Elbow Pitch and Wrist Pitch. Thus, with the base frame origin setup according to the graphical X, Y and Z coordinate system, we are able to represent the manipulator arm as shown in figure 9.

This system possesses the following Denavit-Hartenburg parameters (summarized in Table 4)

Table 4. The Merlin 6500 Vertical Plane Representation and the Corresponding Denavit-Hartenburg Parameters.

i	α_{i-1}	a_{i-1}	d_i	θ_i and range
2	$\alpha_0 = 0^\circ$	$a_0 = 0"$	$d_2 = 0"$	$+ 236^\circ$ to -56°
3	$\alpha_2 = 0^\circ$	$a_2 = 17.38"$	$d_3 = 0"$	$+ 146^\circ$ to -146°
5	$\alpha_4 = 0^\circ$	$a_4 = 17.24"$	$d_5 = 0"$	$+ 90^\circ$ to $- 90^\circ$

Now, using the general form of ${}^{i-1}_{i}T$, given by (4.6), we have

$$ {}^{0}_{2}T = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.9} $$

$$ {}^{2}_{3}T = \begin{bmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.10} $$

and

$$ {}^{3}_{5}T = \begin{bmatrix} c_5 & -s_5 & 0 & a_4 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.11} $$

Using the principle of transformation matrix concatenation, we have

$$ {}^{0}_{3}T = {}^{0}_{2}T \cdot {}^{2}_{3}T \qquad (6.12) $$

which results in

$$ {}^{0}_{3}T = \begin{bmatrix} c_{23} & -s_{23} & 0 & a_2c_2 \\ s_{22} & c_{23} & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6.13) $$

We extract the X- and Y- coordinate positions of {3} from the position vector of (6.13) to get

$$ \left.\begin{aligned} B_x &= a_2\, c_2 \qquad \text{and} \\ B_y &= a_2\, s_2 \end{aligned}\right] \qquad (6.14) $$

Further, since we have

$$ {}^{0}_{5}T = {}^{0}_{3}T \cdot {}^{3}_{5}T \qquad (6.15) $$

we get

$$ {}^{0}_{5}T = \begin{bmatrix} c_{235} & -s_{235} & 0 & a_2c_2 + a_4c_{23} \\ s_{235} & c_{235} & 0 & a_2s_2 + a_4s_{23} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6.16) $$

From (6.16), we extract the position of C, given by

$$\left.\begin{aligned} C_x &= a_2c_2 + a_4c_{23} \\ C_y &= a_2s_2 + a_4s_{23} \end{aligned}\right] \qquad (6.17)$$

Since {D} is d_6" away from {5}, along the normal vector (X_5), the position of D can be computed as

$$D_x = C_x + d_6 \cdot c_{235}$$
$$D_y = C_y + d_6 \cdot s_{235}$$

i.e.

$$\left.\begin{aligned} D_x &= a_2c_2 + a_4c_{23} + d_6c_{235} \\ D_y &= a_2s_2 + a_4s_{23} + d_6s_{235} \end{aligned}\right] \qquad (6.18)$$

Since we now know all the endpoint positions, we can develop the workspace of the manipulator in two planes (the horizontal and the vertical), given the range of motion of each joint. A computer simulation of each of the planar workspaces of the Merlin 6500 manipulator workspace was performed and the results are shown in figures 10 and 11. The source code listings used to generate the vertical and horizontal plane workspaces (using Fortran and the DISSPLA graphics package) are given in Appendix V.

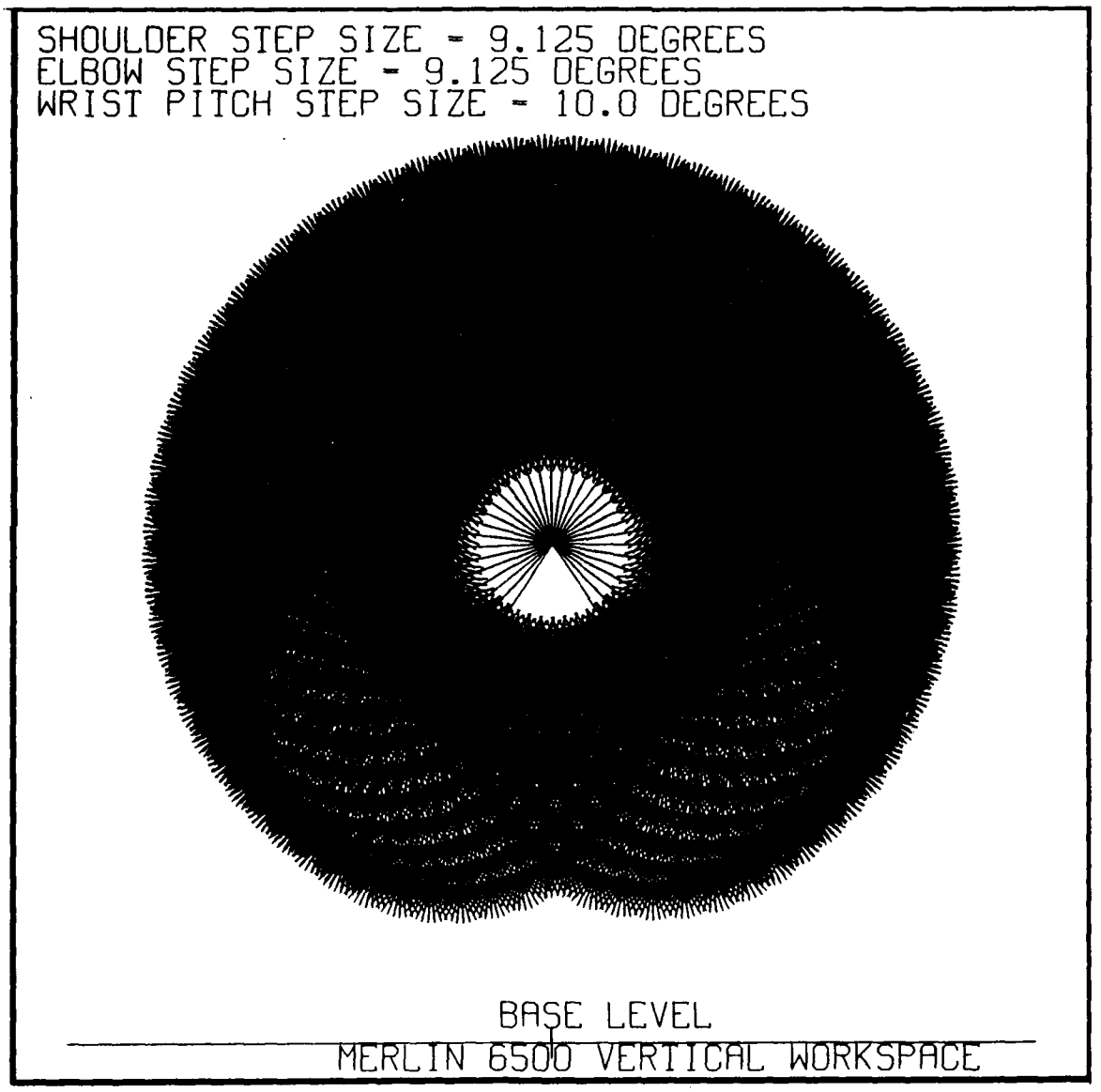


Figure 10. Computer Simulation Results of the Merlin 6500 Vertical Plane Workspace.

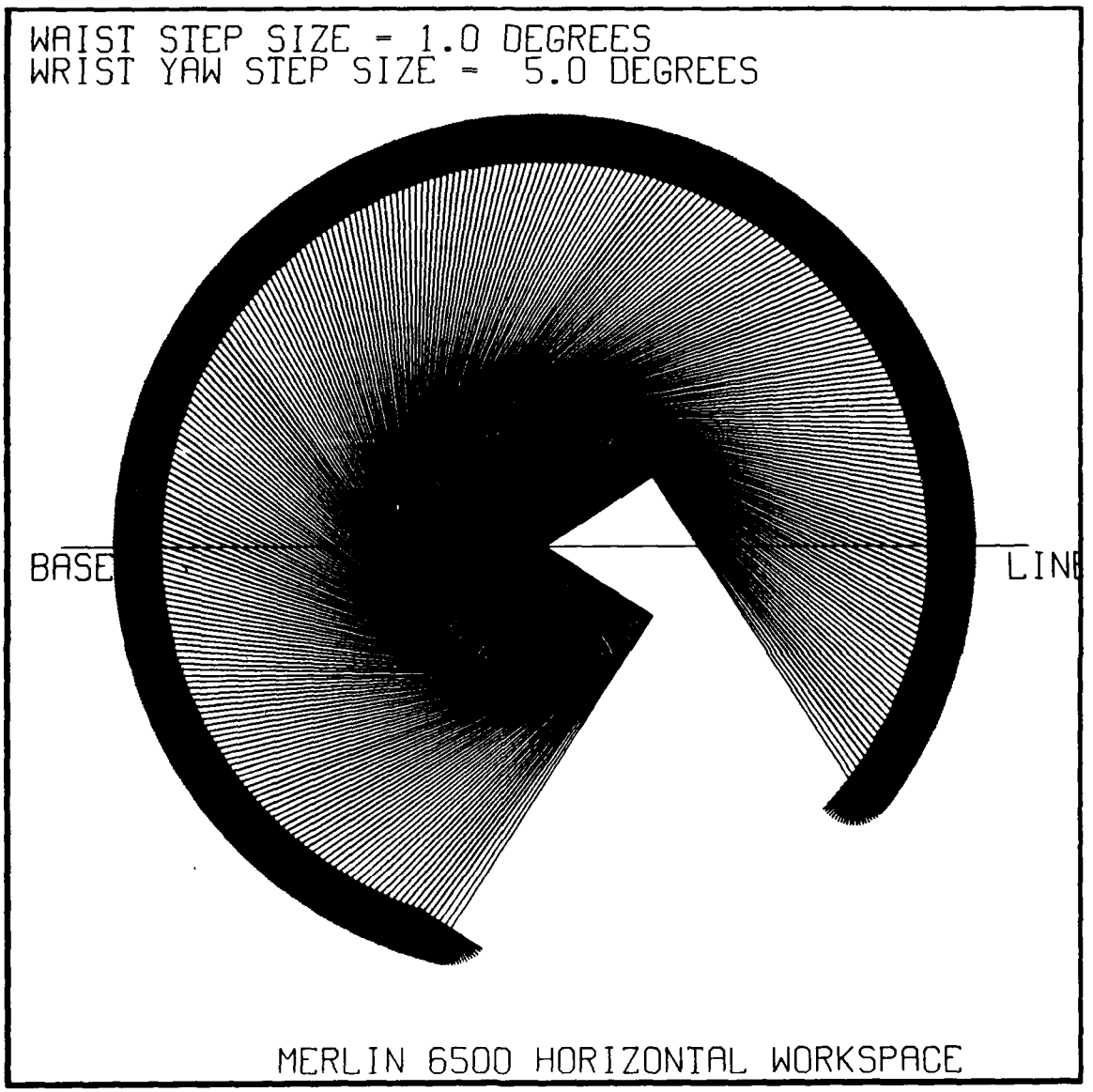


Figure 11. Computer Simulation Results of the Merlin 6500 Horizontal Plane Workspace.

VII THE UTAH/MIT DEXTEROUS HAND

Previous work

The kinematics of articulated hands has been examined in detail previously[12], while that of the Utah/MIT dexterous hand has been solved by Narasimhan[13]. The present work differs from that performed at MIT in that the frame assignments have been performed with a base frame set up at the intersection of the 0^{th} joint axis for the middle finger and the thumb. Further, the origin of the frames for joint 0 of each finger has been located at the intersection of the axis of joint 0 with the a_i perpendicular to axis 0 and which passes through joint 1. The process of direct kinematic closed-form equation development for the dexterous left- and right- hand has also been presented in detail. The current work has been performed keeping in mind the fact that the Utah/MIT dexterous hands have to be attached to manipulators for dexterous tele-operation purposes. Further, the current work proceeds to examine the differences in the direct kinematics of the left- and right-dexterous hands and proposes a minimal-change method for solving the direct kinematics of the right hand, using the closed-form equations of the left hand. The major advantage of such a method is that the direct kinematics of one dexterous hand only has to be programmed, since changes to the values of the appropriate Denavit-Hartenburg parameter values will allow for switching from the left to the right hand, and vice-versa.

Direct Kinematics

The generalized process of direct kinematic closed-form equation development has been dealt with in Chapters 4 and 5. The frame assignments for the left-fingered Utah/MIT dexterous hand is as shown in figures 12 and 13, and follows the basic frame assignment procedure outlined in Chapter 4.

The direct kinematic analysis of the Utah/MIT dexterous hand involves the division of the hand into four manipulators, viz. the thumb, finger 1, finger 2 and finger 3. Finger 1 denotes the finger situated on the thumb side of the palm, finger 2 the middle finger and

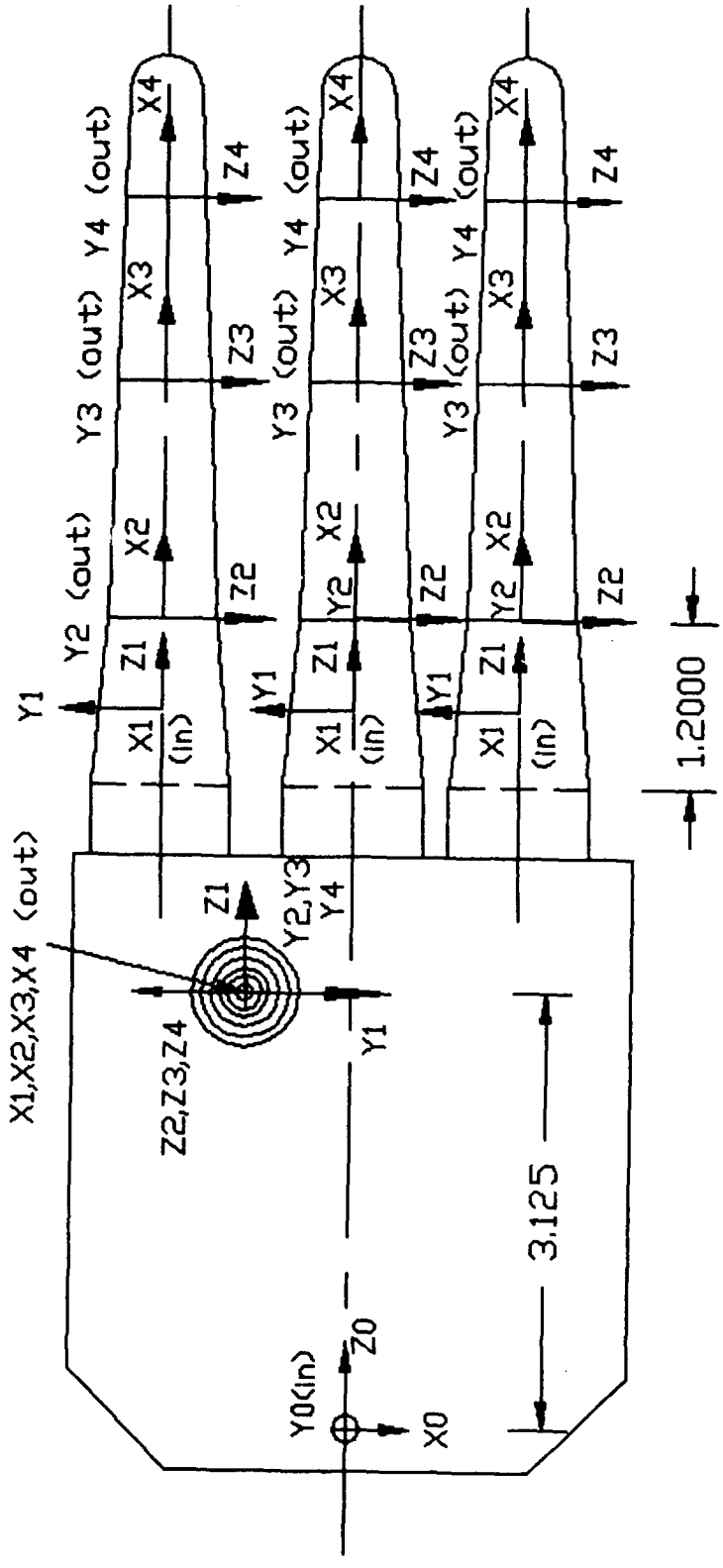


Figure 12. The Utah/MIT Dexterous Hands - Top View

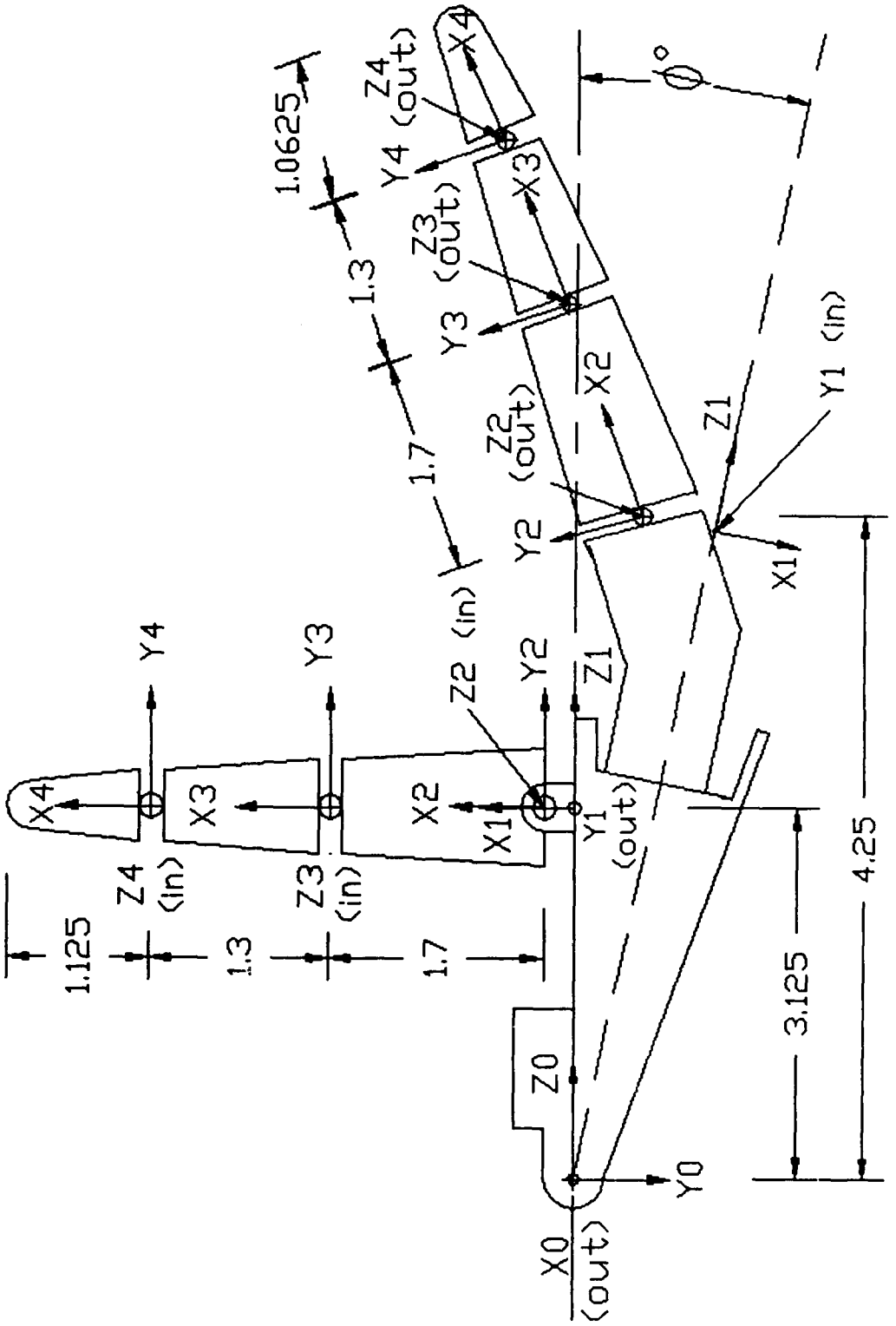


Figure 13. The Utah/MIT Dexterous Hands
- Side View

finger 3 the finger situated on the opposite side of the palm from the thumb. Since the fingers 1, 2 & 3 are kinematically similar with respect to the base frame {0}, except for the offset along X_0, i.e. a_0, we will let a_0 be a variable depending on the finger we are referring to.

Further, since we are dealing with a multiple manipulator system, a trailing sub-script 't' will be used to refer to the thumb while 'f' (f = 1, 2 or 3) will be used to refer to the fingers 1, 2 or 3.

Thumb Kinematics

The direct kinematic closed-form equations for the thumb are now developed with respect to the common base frame.

Denavit-Hartenburg Parameters for the Thumb

We now develop the Denavit-Hartenburg parameters for the thumb, following the rules developed in Chapter 4. The joint variables for each joint are the joint angles, θ_i (i = 1 to 4). We have again adopted the right-hand rule (counter-clockwise is positive) for determining the sign of the angles.

Table 5. The Denavit-Hartenburg Parameters for the Thumb of the Utah/MIT Hand.

i	α_{i-1} (degrees)	a_{i-1} (inches)	d_i (inches)	Joint Variable θ_i (degrees)	Kinematic Range of θ_i (degrees) (A7, 14, 15)
1	$\alpha_1 = 0^\circ$	$a_1 = -0.75"$	$d_2 = 3.125"$	θ_1°	-45° to -135°
2	$\alpha_2 = 90^\circ$	$a_2 = 0.375"$	$d_2 = 0"$	θ_2°	-15° to $+60^\circ$
3	$\alpha_2 = 0^\circ$	$a_2 = 1.7"$	$d_3 = 0"$	θ_3°	$+6.5^\circ$ to $+90^\circ$
4	$\alpha_3 = 0^\circ$	$a_3 = 1.3"$	$d_4 = 0"$	θ_4°	0° to $+90^\circ$

Thumb Transformation Matrices:

The thumb transformation matrices are developed using the ${}^{i-1}_{i}T$ matrix developed in Chapter 4 and is given by equation (4.6).

$$ {}^{0}_{1}T_t = \begin{bmatrix} c_1 & -s_1 & 0 & a_0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Since } a_0 = 0^\circ \qquad (7.1) $$

$$ {}^{1}_{2}T_t = \begin{bmatrix} c_2 & -s_2 & 0 & a_1 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Since } a_1 = 90^\circ \text{ and } d_2 = 0" \qquad (7.2) $$

$$ {}^{2}_{3}T_t = \begin{bmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Since } a_2 = 0^\circ \text{ and } d_3 = 0" \qquad (7.3) $$

$$ {}^{3}_{4}T_t = \begin{bmatrix} c_4 & -s_4 & 0 & a_3 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Since } a_3 = 0^\circ \text{ and } d_4 = 0" \qquad (7.4) $$

Direct Kinematic Equation Development for the Thumb

For the thumb, the direct kinematic closed-form equations can be developed using

$$ {}^{0}_{4}T_t = {}^{0}_{1}T_t \cdot {}^{1}_{2}T_t \cdot {}^{2}_{3}T_t \cdot {}^{3}_{4}T_t \qquad (7.5) $$

Now

$$ {}^{0}_{4}T_t = {}^{0}_{1}T_t \cdot {}^{1}_{2}T_t \cdot {}^{2}_{4}T_t \qquad (7.6) $$

where

$$ {}^{2}_{4}T_{t} = {}^{2}_{3}T_{t} \cdot {}^{3}_{4}T_{t} \tag{7.7} $$

From (7.7), we have

$$ {}^{2}_{4}T_{t} = \begin{bmatrix} c_{34} & -s_{34} & 0 & a_2 + a_3c_3 \\ s_{34} & c_{34} & 0 & a_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Further, from (7.6),

$$ {}^{0}_{4}T_{t} = {}^{0}_{1}T_{t} \cdot {}^{1}_{4}T_{t} \tag{7.8} $$

where

$$ {}^{1}_{4}T_{t} = {}^{1}_{2}T_{t} \cdot {}^{2}_{4}T_{t} \tag{7.9} $$

From (7.9), we have:

$$ {}^{1}_{4}T_{t} = \begin{bmatrix} c_{234} & -s_{234} & 0 & a_1+a_2c_2+a_3c_{23} \\ 0 & 0 & -1 & 0 \\ s_{234} & c_{234} & 0 & a_2s_2+a_3s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.10} $$

From (7.8) we have

$$ {}^{0}_{4}T_{t} = \begin{bmatrix} c_1c_{234} & -c_1s_{234} & s_1 & a_0+c_1(a_1+a_2c_2+a_3c_{23}) \\ s_1c_{234} & -s_1s_{234} & -c_1 & s_1(a_1+a_2c_2+a_3c_{23}) \\ s_{234} & c_{234} & 0 & a_2s_2+a_3s_{23}+d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.11} $$

The above matrix represents the position of the origin of {4} with respect to the base frame {0} and the orientation of the last link with respect to the base frame {0}.

Position of Points on the Last Link

The position of any point on the last link of the thumb which is "L" inches from the origin of frame {4}, is given by

$$
\begin{aligned}
P_x' &= P_x + n_x \cdot L \quad \text{where} \quad p_x = {}^0_4T_t(1,4) \quad \text{and} \quad n_x = {}^0_4T_t(1,1) \\
P_y' &= P_y + n_y \cdot L \quad \text{where} \quad p_y = {}^0_4T_t(2,4), \text{ and } n_y = {}^0_4T_t(2,1) \\
P_z' &= P_z + n_z \cdot L \quad \text{where} \quad p_z = {}^0_4T_t(3,4), \text{ and } n_z = {}^0_4T_t(3,1)
\end{aligned}
\qquad (7.12)
$$

Finger Kinematics

The direct kinematics for the fingers are now developed with respect to the common base frame.

Denavit-Hartenburg Parameters for the Fingers

We now proceed to develop the Denavit-Hartenburg parameters for the fingers of the Utah/MIT dexterous hand. We develop these parameters using the convention outlined in Chapter 4. We have also adopted the right-hand rule (counter-clockwise positive) for determining the sign of angles.

Here, a_i depends on the finger we are referring to. Thus, a_i = -1.375" for finger 1, a_i = 0" for finger 2 and a_i = 1.1875" for finger 3 for the left hand.

It must be mentioned here that the frame assignments are similar for all the fingers, and so the kinematic parameters for the different fingers vary in one regard only, viz. a_o - the offset along the palmar surface.

Table 6. The Denavit-Hartenburg Parameters for the Fingers of the Utah/MIT Hand.

i	α_{i-1} (degrees)	a_{i-1} (inches)	d_i (inches)	Joint Variable θ_i (degrees)	Kinematic Range of θ_i (degrees) [7, 14, 15]
1	$\alpha_1 = -\phi^\circ = -12^\circ$	$a_1 = \begin{matrix} -1.375" \\ 0" \\ 1.1875" \end{matrix}$	$d_2 = \frac{4.25}{\cos\phi} + 1.2\cos 30^\circ$	θ_1	$+65^\circ$ to $+115^\circ$
2	$\alpha_2 = 90^\circ$	$a_1 = -0.6"$	$d_2 = 0"$	θ_2	$+120^\circ$ to $+191^\circ$
3	$\alpha_2 = 0^\circ$	$a_2 = 1.7"$	$d_3 = 0"$	θ_3	$+3.5^\circ$ to $+90^\circ$
4	$\alpha_3 = 0^\circ$	$a_3 = 1.3"$	$d_4 = 0"$	θ_4	0° to $+90^\circ$

Finger Transformation Matrices:

The finger transformation matrices are obtained using the general form of the ${}^{i-1}_{i}T$ matrix developed in Chapter 4 and given by (4.6).

$$ {}^{0}_{1}T_f = \begin{bmatrix} c_1 & -s_1 & 0 & a_{o_f} \\ c_\phi s_1 & c_\phi c_1 & s_\phi & s_\phi d_1 \\ -s_\phi s_1 & -s_\phi c_1 & c_\phi & c_\phi d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Since } a_o = -\phi \ (= -12^\circ), \ a_o = a_{o_f} \ (f = 1, 2 \text{ or } 3) \qquad (7.13) $$

$$ {}^{1}_{2}T_f = \begin{bmatrix} c_2 & -s_2 & 0 & a_1 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Since } \alpha_1 = 90^\circ \text{ and } d_2 = 0" \qquad (7.14) $$

$$ {}^{2}_{3}T_f = \begin{bmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Since } \alpha_2 = 0^\circ \text{ and } d_3 = 0" \qquad (7.15) $$

$$ {}^{3}_{4}T_f = \begin{bmatrix} c_4 & -s_4 & 0 & a_3 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Since } \alpha_3 = 0^\circ \text{ and } d_4 = 0" \qquad (7.16) $$

Direct Kinematic Equation Development for the Fingers

The closed-form direct kinematic equations for the fingers can be developed using

$$ {}^{0}_{4}T_f = {}^{0}_{1}T_f \cdot {}^{1}_{2}T_f \cdot {}^{2}_{3}T_f \cdot {}^{3}_{4}T_f \qquad (7.17) $$

Using the process of transformation matrix concatenation, we have

$$ {}^{0}_{4}T_f = {}^{0}_{1}T_f \cdot {}^{1}_{2}T_f \cdot {}^{2}_{4}T_f \qquad (7.18) $$

where

$$ {}^{2}_{4}T_f = {}^{2}_{3}T_f \cdot {}^{3}_{4}T_f \qquad (7.19) $$

From (7.19), we solve for ${}^{2}_{4}T_f$ as

$$ {}^{2}_{4}T_f = \begin{bmatrix} c_{34} & -s_{34} & 0 & a_2+a_3c_3 \\ s_{34} & c_{34} & 0 & a_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

From (7.18), we can write

$$ {}^{0}_{4}T_f = {}^{0}_{1}T_f \cdot {}^{1}_{4}T_f \qquad (7.20) $$

where

$$ {}^{1}_{4}T_f = {}^{1}_{2}T_f \cdot {}^{2}_{4}T_f \qquad (7.21) $$

We therefore compute ${}^{1}_{4}T_{f}$ using (7.21), to get

$$ {}^{1}_{4}T_{f} = \begin{bmatrix} c_{234} & -s_{234} & 0 & a_1+a_2c_2+a_3c_{23} \\ 0 & 0 & -1 & 0 \\ s_{234} & c_{234} & 0 & a_2s_2+a_3s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (7.22) $$

Again, from (7.20), we get

$$ T_f = \begin{bmatrix} c_1c_{234} & -c_1s_{234} & s_1 & a_{o_f}+c_1(a_1+a_2c_2+a_3c_{23}) \\ c_\phi s_1c_{234} + s_\phi s_{234} & -c_\phi s_1s_{234} + s_\phi c_{234} & -c_\phi c_1 & s_\phi d_1 + c_\phi s_1(a_1+a_2c_2+a_3c_{23}) + s_\phi(a_2s_2+a_3s_{23}) \\ -s_\phi s_1c_{234} + c_\phi s_{234} & s_\phi s_1s_{234} + c_\phi c_{234} & s_\phi c_1 & c_\phi d_1 - s_\phi s_1(a_1+a_2c_2+a_3c_{23}) + c_\phi(a_2s_2+a_3s_{23}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (7.23) $$

The above matrix represents the position of the origin of {4} and the orientation of the last link of the fingers with respect to the base frame {0}.

Positions of Points on the Last Link

The position p' of any point on the last link of the fingers which is 'L' inches from the origin of frame {4} is given by

$$ \begin{aligned} p'_x &= p_x + n_x \cdot L \quad \text{where } p_x = {}^{0}_{4}T_f(1,4) \text{ and } n_x = {}^{0}_{4}T_f(1,1) \\ p'_y &= p_y + n_y \cdot L \quad \text{where } p_y = {}^{0}_{4}T_f(2,4) \text{ and } n_y = {}^{0}_{4}T_f(2,1) \\ p'_z &= p_z + n_z \cdot L \quad \text{where } p_z = {}^{0}_{4}T_f(3,4) \text{ and } n_z = {}^{0}_{4}T_f(3,1) \end{aligned} \qquad (7.24) $$

Equations (7.11), (7.12), (7.23) and (7.24) represent the direct kinematics of the Utah/MIT dexterous left hand.

The Direct Kinematics of the Utah/MIT Right Hand

For the right Utah/MIT dexterous hand, the direct kinematics can be developed in one of two possible ways, viz. by going through the procedure outlined above, with new frame assignments for the right hand, or by using the frames assigned previously and the equations developed above and adjusting the numeric values of the Denavit-Hartenburg parameters to correspond to a right hand.

We have adopted the latter procedure, and have developed and simulated the direct kinematics of the right hand by changing the sign of the a_0 parameter for the thumb and fingers. Thus, for the thumb,

$$a_0 = + 0.75" \tag{7.25}$$

while for the fingers, the value of a_0 is given by

$$a_{0_f} = \begin{cases} 1.375" & \text{for finger 1} \\ 0" & \text{for finger 2} \\ -1.1875" & \text{for finger 3} \end{cases} \tag{7.26}$$

The above adjusted values of the parameters can be used with the direct kinematic equations given by (7.23) to obtain the direct kinematics of the right Utah/MIT dexterous hand.

VIII THE SIMULATION PROGRAM

Introduction

The objective of this computer graphical simulation is to model the kinematic behavior of the Merlin 6500 robot arm and the Utah/MIT dexterous hand, when joined together in user-defined combinations. The aim is to allow a user to simulate and study different kinematic joint arrangements between the Merlin robot and the Utah hand and move the combined systems together in three dimensional space. This should provide the user with the capability of investigating various strategies for physically attaching the Utah/MIT hand to the end of the arm, in terms of the ranges of motion and also the total workspace of the combined system.

Robot Simulation

The simulation adopts the methodology of depicting links by six-sided figures, thus closely approximating the actual system. Each of the links consist of a system, and are connected at the ends to other links (or systems) by joints. All of the joints of the Utah/MIT dexterous hand (sixteen) and the Merlin manipulator (six) are revolute in nature.

The Link Dimensioning Approach

The method used to graphically depict a robot involves breaking the subject robot into sub-units (links) and modeling these sub-units using a six-sided (cubic) figure by using the geometrical spatial relationships of each link's corners.

It is assumed that each link can be represented by a six sided box. The box is dimensioned to be able to closely contain the dimensions of the real link being represented. It thus takes eight points to describe a link. Picture-perfect accuracy is sacrificed with this method, but accurate link orientations can be achieved. The points are measured vectorially with respect to a local origin which is the point of rotation or translation for that particular link of the manipulator. Each link of a robot is related to other links by means of translation and rotation vectors which are defined with respect to the local origin

of the link. The point, translation and rotation vectors can be be stored in a data file that holds all the information necessary to model a manipulator.

Data Files

The eight point vectors defining a link are stored in a specific manner. The first row of eight numbers defines the X-coordinates of the eight points of the link. The second row defines the Y-coordinates, while the third row defines the Z-coordinates. Thus a 3 x 8 matrix is needed to represent each link. For example, a link may be defined as in Table 7.

Table 7. Link Definition in Data Files

Corner number → / Axes ↓	1	2	3	4	5	6	7	8
X-coordinates	8	8	8	8	0	0	0	0
Y-coordinates	2	2	-2	-2	2	2	-2	-2
Z-coordinates	2	-2	2	-2	2	-2	2	-2

Rotation and translation vectors are also stored in a specific manner. The first row is the X, Y and Z coordinates of the translation vector while the second row is the rotation vector, consisting of rotation angles about the X, Y and Z axes. An example of the rotation and translation vectors, as stored in the data file, is given in Table 8.

Once the dimensions for each link of a system is stored in a data file in a form that can be read into the program, the system is defined for the purposes of modeling. Point vectors are stored in order of link sequence as one group, while translation and rotation vectors are stored in order of link sequence as another group. In the main program, arrays are dimensioned to store the dimensional values of a link. A data point

Table 8. Rotation and Translation Vectors Stored in Data Files

Coordinate	X	Y	Z
Translation vector	30"	0"	5"
Rotation angles	90°	0°	0°

array (an example is POINTS) is dimensioned as POINTS(8,3,I,J), where 8 denotes the number of points in each link, 3 the number of coordinate dimensions, I the number of links in the system and J the number of systems. The translational/ rotational array (an example is GO) could be dimensioned as GO(3,I,J,K), where 3 denotes the number of dimensions, I = 1 the translational vector and I = 2 the rotational vector, J the number of links and K the number of systems.

Link Dimensioning

The robot model is made up of a series of links, for example the base link, stand, waist, upper arm and lower arm. The origin of each link is the point of rotation of the joint. The dimensional relationship between each link must be defined. The notation necessary to relate one link to the previous link is a translation vector and a rotation vector. The link translation vector originates at the previous link in the open kinematic chain and ends at the origin of the current link. It is defined with respect to the previous link's coordinate system. The rotation vector relates the orientation of the current coordinate system to the previous coordinate system.

Rotational transformation must occur in a fixed order, viz. rotation about the X-axis first, followed by rotation about the Y-axis and finally, if necessary, about the Z-axis. (The Z-rotation component is currently left as zero, thereby saving that component for use as a dynamic rotation in the program). For a link with a coordinate system translated 30 units in the X-direction and offset 5 units in the Z-direction, with a 90 degree rotation about the X-axis, the vectors

would be represented in a data file as in table 9.

Table 9. Link Translations and Rotation Angles.

Coordinate	X	Y	Z
Translation	30"	0"	5"
Rotation angles	90°	0°	0°

The first row defines the translational vector components in the X, Y and Z coordinate directions and the second row defines the rotation angles about the X, Y and Z axes. These values are used to fill the transformation matrix for a specific link.

Coordinate System Transformation

The orientation of each link is defined with respect to the previous link using rotation and translation vectors. The values of the vectors are measured with respect to the previous coordinate system. The vectors are loaded into a transformation matrix by using input data from the matrix [GO] to form the transformation matrix [TR]. The following equations are used to determine each component of the matrix [TR]

```
GO(1,1,L,S) = Translation in the X direction
              (L is the link number and S is the system number)
GO(2,1,L,S) = Translation in the Y direction
GO(3,1,L,S) = Translation in the Z direction
GO(1,2,L,S) = Rotation about the X axis
              (Counterclockwise is deemed positive),
GO(2,2,L,S) = Rotation about the Y axis
GO(3,2,L,S) = Rotation about the Z axis
```

The transformation matrix, TR, is formed using :

```
TR(1,1,L,S) = COS(GO(3,2,L,S)) * COS(GO(2,2,L,S))
TR(1,2,L,S) = -SIN(GO(3,2,L,S)) * COS(GO(2,2,L,S))
```

```
TR(1,3,L,S) = SIN(GO(2,2,L,S))
TR(1,4,L,S) = GO(1,1,L,S)
TR(2,1,L,S) = COS(GO(3,2,L,S)) * SIN(GO(2,2,L,S)) *
         SIN(GO(1,2,L,S)) + SIN(GO(3,2,L,S)) * COS(GO(1,2,L,S))
TR(2,2,L,S) = -SIN(GO(3,2,L,S)) * SIN(GO(2,2,L,S)) *
         SIN(GO(1,2,L,S)) + COS(GO(3,2,L,S)) * COS(GO(1,2,L,S))
TR(2,3,L,S) = -COS(GO(2,2,L,S)) * SIN(GO(1,2,L,S))
TR(2,4,L,S) = GO(2,1,L,S)
TR(3,1,L,S) = -COS(GO(3,2,L,S)) * SIN(GO(2,2,L,S)) *
         COS(GO(1,2,L,S)) + SIN(GO(3,2,L,S)) * SIN(GO(1,2,L,S))
TR(3,2,L,S) = SIN(GO(3,2,L,S)) * SIN(GO(2,2,L,S)) *
         COS(GO(1,2,L,S)) + COS(GO(3,2,L,S)) * SIN(GO(1,2,L,S))
TR(3,3,L,S) = COS(GO(1,2,L,S)) * COS(GO(2,2,L,S))
TR(3,4,L,S) = GO(3,1,L,S)
TR(4,1,L,S) = 0
TR(4,2,L,S) = 0
TR(4,3,L,S) = 0
TR(4,4,L,S) = 1
```

The notation used for transformation from {B} to {A} is BTA (B transformed to A). To describe points in {C} with respect to {A}, the transformation matrix BTA must be premultiplied by the transformation matrix CTB. This operation is performed in each link so that points in each link can be defined in the base or global coordinate system. We need to know the definition of the coordinates of a point defined in {C} in the coordinate system of {A}. To find this, it is necessary to multiply the transformation matrix CTA by the vector defining the point in {C}.

To get the vector coordinates of a point in a link in the global system, the (4X4) transformation matrix is multiplied by a (4X1) vector composed of the three coordinates of the point and a 1 in the fourth row. The result of the product is the definition of the vector in the global system. This operation has to be performed for all links in all systems.

Once the dimensions of each point of each link are known in the global system, the three dimensional points must be transformed into two

dimensional screen drawing points.

A perspective view is desired, where objects appear to be shrinking with increasing distance. This requires choosing a focal point, defined in global X and Y coordinates, and a viewing point (or VPOINT - a distance along the global Z axis).

A drawn view really has tunnel vision, it can only 'see' objects that are within a 20 degree cone directly in front of the viewer. Imagine two lines, one drawn from the viewpoint to the global origin and the other drawn from the viewpoint to a point defining a link. If this angle between these two lines is zero, the point will be assigned 2-D coordinates of (0,0). If the angle is equal to 20 degrees, the 2-D coordinates will be assigned coordinates corresponding to the edge of the screen. If the angle lies between 0 and 20 degrees, the assigned 2-D coordinates will be assigned proportional to the angle. If the angle is greater than 20 degrees, then the point will not be shown in the view.

Once the 2-D coordinates of all points are known, a graphics package can be employed to connect lines between the appropriate points so that the cubes defining the links can be drawn. The robot thus consists of a series of links assembled together at the joints.

The Graphics Software Menus

The graphics package operates using a VT 240 or a Tektronix 4010 screen. It is organized to be user-friendly, and thus incorporates a main menu and sub-menus. The organization of the menus and sub-menus follows a logical pattern determined by user operations. The program begins with a main menu and two sub-menus. The main menu is the entry point for both the sub-menus as well as the program exit point. One of the options in the main menu also allows the user to see the system, as currently defined, on the screen.

Major sub-menus consist of the setup and operations menus. The set-up sub-menu allows the user to perform the tasks of defining the system in terms of the viewpoint, the focal point and the factor of magnification for subsequent views, which of the possible systems (the room, left/right arm Merlin, left/right Utah/MIT dexterous hand) are to be drawn, the relative position of the Utah/MIT hands with respect to

the end of the Merlin arms, the positioning of the robots in the room, drawing the systems as currently defined, which of the two possible remote slave systems is to be currently active (only one slave system is active at any one time) and finally, returning to the main menu.

The Execution sub-menu allows a user to move the individual joints of the selected systems, viz. the Merlin and the Utah/MIT hand, to save a view and to recall a saved view, to move the Merlin from its current position to another point, defined by its position and orientation, to move all joints of the robot, in user-defined steps, and to return to the main menu.

Each item in each of the menus is discussed in more detail below.

The Main Menu

The user chooses one of the following options from the main menu

Go to the setup menu.
Go to the execution menu.
Draw the system, as currently defined.
Exit from the program.

The Setup Menu

The set-up menu consists of the following options, each of which is explained below :

VIEWPOINT

The position in global coordinates the the robot is to be viewed from is chosen by this operation. The global origin is located at the lower, far left corner of the room. On the screen, 'X' is positive towards the right, 'Y' is positive towards the top and 'Z' is positive coming out of the screen.

FOCUS AND MAGNIFICATION

The focal point is defined in screen coordinates as (0,0) and is in the center of the screen. The focal point is not in global coordinates, which factor may cause confusion when the viewpoint is changed and the focal point is not (0,0). Each view, no matter what the viewpoint, has

a unique screen focal point, i.e. (0,0). Any new focal point for a specific view is measured with respect to the focal point at (0,0).

The user may define the magnification for a specific view. Increasing the magnification will make the objects being viewed appear larger. A point in the center of the screen will remain in the center of the screen.

SYSTEMS DRAWING

The following are defined as systems:

The Room in which the slave systems are placed,
A Left-Arm Merlin Robot,
A Left Utah/MIT dexterous hand,
A Right-Arm Merlin Robot,
A Right Utah/MIT dexterous hand.

Prior to any specific view, the user may choose to enable or disable the drawing of any of the above systems. After selecting this option, a menu is displayed, prompting the user to choose one of the available options related to drawing (or not drawing) a system.

FIXED HAND POSITIONING

The fixed position of the Utah/MIT hand relative to the Merlin wrist may be changed by the user. The base coordinate system of the hand is related to the wrist coordinate system by a translation vector and a 90 degree rotation about the 'X' axis of the wrist coordinate system. The hand can be positioned with respect to the robots hand roll system by a translation vector and a rotation vector. After selecting this option, the user is prompted to define these vectors.

REPOSITION ROBOT

The robot may be positioned anywhere in the room. The origin of the room coordinate system is the far left, bottom corner of the room as seen in the initial view. The position of the robot is defined by a vector from the origin of the room to the center of the base of the robot. The program prompts the user for input necessary in redefining this vector.

CHANGE ACTIVE ROBOT

Robot positioning and reconfiguring is performed working from the menus during the operation of the program. It is possible for two robots to be viewed at the same time. However, only one robot is deemed active at any one time. This option allows the user to choose which robot is the one to be acted upon at any one time. This allows for multiple arms to be used in simulations.

RETURN TO MAIN MENU

Choosing this option returns the user to the main menu.

DRAW ROBOT

This option provides a view of the systems, as currently configured.

The Execution Menu

The execution menu possesses several options, each of which are discussed below :

MOVE ROBOT JOINT ANGLES

Each of the joint angles of the Merlin robot can be moved individually. Once this option is chosen, another menu will be displayed prompting the user to choose the link to be repositioned. Once a link is chosen, the user is informed of the current angle and is prompted for the desired joint angle in degrees. After input, the link menu is displayed again until the user requests an exit from that menu.

MOVE INDIVIDUAL FINGER JOINTS

Each of the Utah/MIT hand's joint angles can be moved individually. Once this option is chosen, another menu is displayed prompting the user to choose the finger to be repositioned. Once a finger is chosen, a third menu is displayed prompting the user to choose the joint to be repositioned. Next the user is informed of the current specified joint angle and is prompted for the desired joint angle in degrees. After input, the joint menu is displayed again until the user requests to exit that menu, following which the finger-choosing menu is displayed until

an exit is requested.

SAVE THIS VIEW

If a user wishes to leave his work and resume later at the ending point of the last session, this option will store the current parameters that define the configuration of the present view. These parameters are written to a file named 'SAVE.DAT'. Only one view can be stored during the course of a program run. A second save will write over the first save. After leaving the program, the save file could be renamed to avoid being written over by a future run. If this renamed file is to be used in a program, it will need to be copied to 'SAVE.DAT' prior to running the program.

DRAW ROBOT FROM SAVED DATA FILE

The user may resume work from a previously saved parameter file. After selecting this option, the next view drawn will be defined by parameters read from a file named 'SAVE.DAT'.

MOVE ROBOT TIP POINT TO POINT

This option uses the inverse kinematics of the Merlin robot arm to move the tip of the Merlin arm from the current position to the user defined position. The goal (or desired) position is chosen by specifying the desired position and orientation of the tip of the Merlin manipulator, relative to the global frame of the Merlin (defined in Chapter 5). The user also has to select the desired joint angles from the computed set of valid angles that are determined by the inverse kinematic computation. The file INKIN.FOR has to be compiled and linked to the simulation program for this option to be active.

MOVE ALL JOINTS OF THE ROBOT, IN STEPS

The user will be prompted to enter a complete set of six joint angles and the number of steps. The user also chooses to either erase between views, or to draw each view on the same screen.

Figure 14 shows a succession of positions, detailing the robot's movements from the current position to the position defined by the set

of angles that were entered by the user. The number of views is determined by the number of steps. Intermediate angles are computed by interpolation of the initial and final angles.

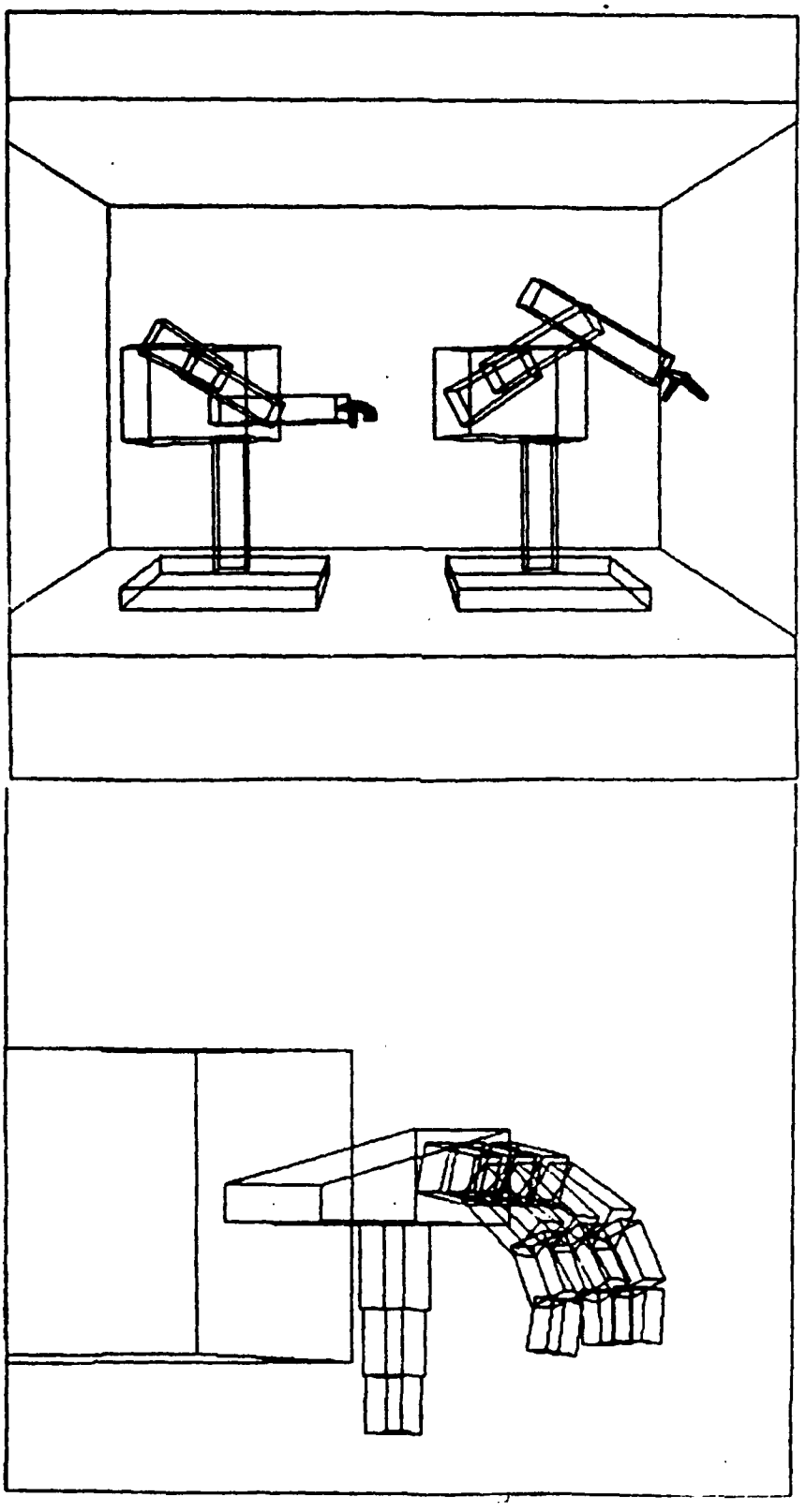


Figure 14. The Computer Graphical Program - Simulation Results.

IX CONCLUSIONS

The Results

This study has presented the methodology and mathematics of kinematic analysis in chapters I, II, III and IV, followed by the derivation of the direct and inverse kinematic closed-form equations for the Merlin 6500 left- and right- shouldered manipulator in chapter V. A graphical simulation proceedure and results of the planar workspace for the Merlin 6500 left-shouldered robot arm is detailed in chapter VI. The study then proceeds to examine the development of the direct kinematic closed-form equations for the Utah/MIT dexterous hand in chapter VII. This is followed by a discussion of the computer graphical simulation for the Merlin and Utah/MIT dexterous hand, when combined in user defined configurations, detailed in chapter VIII.

Further Work

The current study has examined only the direct kinematics of the Utah/MIT dexterous hand. As the slave system is driven from a remote location by a human arm encased in an exo-skeleton, and since it is necessary for the slave system to grasp objects at the same position and with the same orientation as the driving master system, it becomes necessary to perform kinematic transformations between the master and the slave systems. These transformations will typically involve the direct kinematics of the human hand, whose output data can be utilized as an input to the inverse kinematics of the Utah/MIT hand, thus allowing for objects to be grasped by the slave system in a similar fashion to the master system. Thus, a study of the kinematic mapping between the human hand and the Utah/MIT hand must be made, so that when the activating system (the human arm) grasps an object, the remote system follows it in action. Lastly, there is a need to study the mechanism of object grasping by the human hand as well as by the Utah/MIT hand, for different object geometries. This will permit the Utah/MIT hand to grasp and manipulate the remote object in as dexterous a fashion as the human arm.

The current study notes the existence of the remotizer of the Utah/MIT dexterous hand and the constraint it poses for the operation of

the remote system. A further study must also deal with the effect of the remotizer on the use of the Utah/MIT hand as an end-effector for the slave system. This would typically involve studying the effect of the remotizer on the workspace of the remote slave system, such that the workspace would be a maximum without undue effect on the remotizer links.

APPENDICES

APPENDIX 1

DEFINITIONS

In any scientific study, it is necessary to make clear the meaning of certain technical words that are being used, so as to avoid confusion in their use by different users with varying backgrounds. As such, certain key words used in this study are explained below :-

A1.1 Kinematics Kinematics is the science of motion which treats motion without regard to the forces that cause it. Within the science of kinematics, one may study the geometrical properties of motion or the time derivatives of position. We limit the present study to the geometrical properties of motion.

A1.2 Manipulator (or Robot) A manipulator is kinematically defined to be a set of nearly rigid links connected together in a chain by joints which allow relative motion of the neighbouring links. In the case of rotary or revolute joints, the displacements are joint angles, while in the case of sliding or prismatic joints, these displacements are joint offsets.

A1.3 Degrees of freedom The degrees of freedom present at any joint of a mechanism are computable as the number of independent position variables that need to be specified to locate specific parts of the mechanism. In the case of typical industrial manipulators, since such a manipulator is usually an open kinematic chain, and

because each joint position is usually defined with a single variable, the number of joints equals the number of degrees of freedom. Each joint may, however, possess one or more degrees of freedom.

A1.4 Frame A frame is defined to be a co-ordinate system attached to a joint of a manipulator. The end-frame is generally attached to the tip of the manipulator or to the last joint of the open kinematic chain, while the base frame is generally attached to a non-moving component of the manipulator. In this report, a frame is always referenced by the character inside {}.

A1.5 Cartesian space is defined as the space in which the position of a point is given by three position data values along the three orthogonal axes, X, Y and Z, while the orientation of a body is given by three orientation data value sets.

A1.6 Joint Space is defined as the space in which the position of a point, and the orientation of a link, are defined in terms of the joint variable (or degrees of freedom).

A1.7 Forward Kinematics The forward (or direct) kinematic problem is defined to be the computation of the position and orientation of the end-effector frame relative to the base frame. This problem can also be thought of as changing the representation of manipulator position and orientation from a joint space description into a Cartesian space description.

A1.8 Inverse Kinematics The inverse kinematics problem is defined to be the computation of the joint variables when the position and orientation of the end-effector frame is known with respect to the base frame. This problem can also be thought of as changing the representation of manipulator position and orientation from a Cartesian space description to a joint space description.

A1.9 Workspace The work-space of a manipulator is defined as the set of positions which the end-effector can achieve when the joints degrees of freedom vary over the full range of possible values.

A1.10 Reachable work-space The reachable work-space of a manipulator is defined as that volume of space which the robot end-effector can reach in at least one orientation.

A1.11 Dextrous work-space The dexterous work-space of a manipulator is defined as the volume of space which the robot end-effector can reach with all possible orientations. The dexterous work-space of a manipulator is always a sub-set of the reachable work-space of that manipulator.

APPENDIX 2

ROTATION AND TRANSLATION MATRICES

In chapter IV, we derived the general form of the transformation ${}^{i-1}_{i}T$ by using equation (4.1) to (4.5). Equation (4.4) involved rotations and translations about the X_i and Z_i axes. These rotation and translation matrices are shown below, in general terms.

Let us first examine a rotation of α° about the X axis. The transformation matrix is given by

$$\text{Rot}(X, \alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \text{Cos } \alpha & -\text{Sin } \alpha \\ 0 & \text{Sin } \alpha & \text{Cos } \alpha \end{bmatrix}$$

A translation along the X axis by a distance of 'a' is given by

$$\text{Trans}(X, a) = \begin{bmatrix} 0 & 0 & a \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

A rotation about the Z axis by an angle θ° is given by

$$\text{Rot}(Z, \theta) = \begin{bmatrix} \text{Cos } \theta & -\text{Sin } \theta & 0 \\ \text{Sin } \theta & \text{Cos } \theta & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

A translation along the Z axis for a distance 'd' is given by

$$\text{Trans}(Z, d) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & d \end{bmatrix}$$

Therefore, in equation (4.5), if we substitute the appropriate form of the rotations and translations, viz. a rotation about the X_i axis by α_{i-1} degrees and a translation along the X_i axis by a_{i-1} (i.e. a Screw$\{X_i, \alpha_{i-1}, a_{i-1}\}$), and a rotation about Z_i by θ_i degrees and a translation along the Z_i axis by a distance d_i (i.e. a Screw$\{Z_i, \theta_i, d_i\}$), and multiply out equation (4.5), we get equation (4.6), as follows

$$\overset{i-1}{i}T = \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & c\alpha_{i-1} & -s\alpha_{i-1} & 0 \\ 0 & s\alpha_{i-1} & c\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

i.e.

$$\overset{i-1}{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i\, c\alpha_{i-1} & c\theta_i\, c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}\, d_i \\ s\theta_i\, s\alpha_{i-1} & c\theta_i\, s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}\, d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

APPENDIX 3

DIRECT KINEMATICS SIMULATION FOR THE MERLIN 6500 - LEFT ARM

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  MERLIN ROBOT LEFT ARM KINEMATICS SIMULATION PROGRAM
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  MAIN PROGRAM
C  DEFINE REAL & INTEGER VARIABLES
      INTEGER INTRO
      REAL T(4,4),W,S,E,WR,WP,HR
C  DESCRIBE THE PROGRAM
      PRINT *,' THIS PROGRAM PERFORMS A MATHEMATICAL SIMULATION OF'
      PRINT *,' THE KINEMATICS OF THE MERLIN 6500 LEFT ARM ROBOT.'
 10   PRINT *,' '
      PRINT *,' DO YOU NEED AN INTRODUCTION TO THE PROGRAM ? '
      PRINT *,' YES ----> 1.'
      PRINT *,' NO  ----> 2.'
      INTRO = 2
      READ(5,*) INTRO
      IF(INTRO .EQ. 1) THEN
      CALL INTROD
      ELSE IF(INTRO .EQ. 2) THEN
          GOTO 11
      ELSE
      PRINT *,' ENTRY ERROR'
      GOTO 10
      ENDIF
C  FIND THE USER-DEFINED ANGLES
 11   CALL ANGLES(W,S,E,WR,WP,HR)
      PRINT *,' '
      CALL DIRKIN(W,S,E,WR,WP,HR,T)
      CALL TOUT(T)
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  INTRODUCTION TO THE PROGRAMME
      SUBROUTINE INTROD
      PRINT *,' THE PROGRAM REQUESTS THE USER TO ENTER THE JOINT'
      PRINT *,' ANGLES FOR EACH OF THE FOLLOWING JOINTS :--'
      PRINT *,'   JOINT                  RANGE '
      PRINT *,'   -----                  ----- '
      PRINT *,' WAIST JOINT     ( RANGE + 147 TO - 147 DEGREES )'
      PRINT *,' SHOULDER JOINT  ( RANGE +  56 TO - 236 DEGREES )'
      PRINT *,' ELBOW JOINT     ( RANGE +  56 TO - 236 DEGREES )'
      PRINT *,' WRIST ROLL      ( RANGE + 360 TO - 360 DEGREES )'
      PRINT *,' WRIST PITCH     ( RANGE +  90 TO -  90 DEGREES )'
      PRINT *,' TOOL ROLL       ( RANGE + 360 TO - 360 DEGREES )'
      PRINT *,' '
      PRINT *,' THE PROGRAM RETURNS THE FINAL TRANSFORMATION'
      PRINT *,' MATRIX i.e. THE POSITION & ORIENTATION MATRIX'
      PRINT *,' DEFINED AT THE WRIST PIN OR TIP OF THE ARM.'
      PRINT *,' '
```

```
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  ENTRY OF JOINT ANGLES BY THE USER
      SUBROUTINE ANGLES(W,S,E,WR,WP,HR)
      REAL W1,S1,E1,WR1,WP1,HR1,W,S,E,WR,WP,HR,PI
C  DEFINE PI
      PI = 3.141592654
C  WAIST
 101  W = 0.0
      W1 = 0.0
      PRINT *,' ENTER WAIST ANGLE ( +/- 147 DEGREES ) ==> '
      READ(5,*) W1
      IF (ABS(W1) .GT. 147.0) THEN
      PRINT *,' ERROR -- RANGE IS +/- 147 DEGREES. '
        GOTO 101
      ENDIF
      W = W1 * PI / 180.0
C  SHOULDER
 102  S = 0.0
      S1 = 0.0
      PRINT *,' ENTER SHOULDER ANGLE ( 56 TO -236 DEGREES ) ==> '
      READ(5,*) S1
      IF ((S1 .GT. 56.0) .OR. (S1 .LT. -236.0)) THEN
        PRINT *,' ERROR -- RANGE IS 56 TO -236 DEGREES. '
        GOTO 102
      ENDIF
      S = S1 * PI / 180.0
C  ELBOW
 103  E = 0.0
      E1 = 0.0
      PRINT *,' ENTER ELBOW ANGLE ( 56 to -236 DEGREES ) ==> '
      READ(5,*) E1
      IF ((E1 .GT. 56.0) .OR. (E1 .LT. -236.0))THEN
        PRINT *,' ERROR -- RANGE IS 56 TO -236 DEGREES. '
        GOTO 103
      ENDIF
      E = E1 * PI / 180.0
C  WRIST ROLL
 104  WR = 0.0
      WR1 = 0.0
      PRINT *,' ENTER WRIST ROLL ( +/- 360 DEGREES ) ==> '
      READ(5,*) WR1
      IF (ABS(WR1) .GT. 360.0) THEN
        PRINT *,' ERROR -- RANGE IS +/- 360 DEGREES. '
        GOTO 104
      ENDIF
      WR = WR1 * PI / 180.0
C  WRIST PITCH
 105  WP = 0.0
      WP1 = 0.0
      PRINT *,' ENTER WRIST PITCH ( +/- 90 DEGREES) ==> '
      READ(5,*) WP1
      IF (ABS(WP1) .GT. 90.0) THEN
```

```
        PRINT *,' ERROR -- RANGE IS +/- 90 DEGREES. '
        GOTO 105
      ENDIF
      WP = WP1 * PI / 180.0
C  HAND ROLL
 106  HR = 0.0
      HR1 = 0.0
      PRINT *,' ENTER HAND ROLL ( +/- 360 DEGREES ) ==> '
      READ(5,*) HR1
      IF(ABS(HR1) .GT. 360.0) THEN
        PRINT *,' ERROR -- RANGE IS +/- 360 DEGREES. '
        GOTO 106
      ENDIF
      HR = HR1 * PI / 180.0
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  KINEMATICS IMPLEMENTATION
      SUBROUTINE DIRKIN(W,S,E,WR,WP,HR,T)
      INTEGER I,J,TIP
      REAL T(4,4),C1,C2,C3,C4,C5,C6,S1,S2,S3,S4,S5,S6,C23,S23,D2,D3,
     $R11A,R11B,R12A,R12B,R13A,R13B,R14A,R14B,W,S,E,WR,WP,HR,D6
C  INITIALIZE MATRIX
      DO 301 I = 1,4
        DO 301 J = 1,4
          T(I,J) = 0.0
 301  CONTINUE
      T(4,4) = 1.0
C  DEFINE COSINES AND SINES
      C1 = COS(W)
      C2 = COS(S)
      C3 = COS(E)
      C4 = COS(WR)
      C5 = COS(WP)
      C6 = COS(HR)
      S1 = SIN(W)
      S2 = SIN(S)
      S3 = SIN(E)
      S4 = SIN(WR)
      S5 = SIN(WP)
      S6 = SIN(HR)
      C23 = COS(S + E)
      S23 = SIN(S + E)
C  DEFINE D2, D3 & D6( APPROXIMATELY ) FOR THE LEFT ARM.
C  D2 IS THE SIGNED DISTANCE FROM THE GLOBAL X AXIS TO THE SHOULDER
C  CENTRAL AXIS ( ALIGNED ALONG THE UPPER ARM ). D3 IS THE SIGNED
C  DISTANCE FROM THE SHOULDER CENTRAL AXIS TO THE ELBOW CENTRAL
C  AXIS, ALIGNED   WITH THE CENTER OF THE LOWER ARM. THESE VALUES
C  (D2 & D3) CHANGE SIGN FOR THE RIGHT ARM.  D6 IS THE SIGNED
C  DISTANCE FROM THE WRIST PIN TO THE TIP OF THE ROBOT ARM.
C  ( REFER TO ARM KINEMATICS FOR MORE DETAILS. )
      D2 = 19.00
      D3 = -7.00
      D6 = 3.5
```

```
C  DEFINE TRANSFORM MATRIX ENTRIES
      R11A = (C23 * ((C4 * C5 * C6)-(S4 * S6))-(S23 * S5 * C6))
      R11B = ((S4 * C5 * C6) + (C4 * S6))
      T(1,1) = (C1 * R11A) + (S1 * R11B)
      T(2,1) = (S1 * R11A) - (C1 * R11B)
      T(3,1) = -S23 * ((C4 * C5 * C6)-(S4 * S6))-(C23 * S5 * C6)
      R12A = - C23 * ((C4 * C5 * S6)+(S4 * C6))+(S23 * S5 * S6)
      R12B = ((S4 * C5 * S6)-(C4 * C6))
      T(1,2) = (C1 * R12A)-(S1 * R12B)
      T(2,2) = (S1 * R12A)+(C1 * R12B)
      T(3,2) = S23 * ((C4 * C5 * S6)+(S4 * C6))+(C23 * S5 * S6)
      R13A = (C23 * C4 * S5)+(S23 * C5)
      R13B = (S4 * S5)
      T(1,3) = (-C1 * R13A)-(S1 * R13B)
      T(2,3) = (-S1 * R13A)+(C1 * R13B)
      T(3,3) = (S23 * C4 * S5)-(C23 * C5)
      R14A = (-17.24 * S23)+(17.38 * C2)
      R14B = (D2 + D3)
      T(1,4) = (C1 * R14A)-(S1 * R14B)
      T(2,4) = (S1 * R14A)+(C1 * R14B)
      T(3,4) = (-17.24 * C23)-(17.38 * S2)
C  DECIDE ON DATA TO BE REPORTED TO TIP OR WRIST PIN
      TIP = 0
      PRINT *,' DO YOU WANT POSITION TO BE REPORTED TO TIP '
      PRINT *,' OF ROBOT ARM (1) OR WRIST PIN (0) ? '
      READ(5,*) TIP
C  IF POSITION DATA IS TO BE REPORTED W.R.T. TIP OF ROBOT ARM
C  ADD D6 * APPROACH VECTOR TO POSITION VECTOR I.E.
C  T(ROW,4) = T(ROW,4) + (T(ROW,3) * D6)
      IF(TIP .EQ. 1) THEN
        T(1,4) = T(1,4) + (T(1,3) * D6)
        T(2,4) = T(2,4) + (T(2,3) * D6)
        T(3,4) = T(3,4) + (T(3,3) * D6)
      ENDIF
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  OUTPUT TO SCREEN
      SUBROUTINE TOUT(T)
      REAL T(4,4)
      INTEGER I,J
C  OPEN DATA FILE
      OPEN(UNIT=6,STATUS='NEW',FILE='LDKIN.OUT')
      PRINT *,' '
      DO 601 I = 1,4
C  WRITE TO SCREEN
        WRITE(5,*) (T(I,J),J=1,4)
C  WRITE TO OUTPUT FILE LDKIN.OUT
        WRITE(6,*) (T(I,J),J=1,4)
 601  CONTINUE
      PRINT *,' '
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

DIRECT KINEMATICS SIMULATION FOR MERLIN 6500 RIGHT ARM

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  MERLIN ROBOT RIGHT ARM KINEMATICS SIMULATION PROGRAM
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  MAIN PROGRAM
C  DEFINE REAL & INTEGER VARIABLES
      INTEGER INTRO
      REAL T(4,4),W,S,E,WR,WP,HR
C  DESCRIBE THE PROGRAM
      PRINT *,' THIS PROGRAM PERFORMS MATHEMATICAL SIMULATION OF'
      PRINT *,' THE KINEMATICS OF THE MERLIN 6500 RIGHT ARM ROBOT.'
 10   PRINT *,'  '
      PRINT *,' DO YOU NEED AN INTRODUCTION TO THE PROGRAM ? '
      PRINT *,' YES ----> 1.'
      PRINT *,' NO  ----> 2.'
      INTRO = 2
      READ(5,*) INTRO
      IF(INTRO .EQ. 1) THEN
        CALL INTROD
      ELSE IF(INTRO .EQ. 2) THEN
        GOTO 11
      ELSE
        PRINT *,' ENTRY ERROR '
        GOTO 10
      ENDIF
C  FIND THE USER-DEFINED ANGLES
 11   CALL ANGLES(W,S,E,WR,WP,HR)
      PRINT *,'    '
      CALL DIRKIN(W,S,E,WR,WP,HR,T)
      CALL TOUT(T)
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  INTRODUCTION TO THE PROGRAMME
      SUBROUTINE INTROD
      PRINT *,' THE PROGRAM REQUESTS THE USER TO ENTER THE JOINT '
      PRINT *,' ANGLES FOR EACH OF THE FOLLOWING JOINTS :-- '
      PRINT *,'   JOINT                    RANGE '
      PRINT *,'   -----                    ----- '
      PRINT *,' WAIST JOINT     ( RANGE + 147 TO - 147 DEGREES ) '
      PRINT *,' SHOULDER JOINT  ( RANGE +  56 TO - 236 DEGREES ) '
      PRINT *,' ELBOW JOINT     ( RANGE +  56 TO - 236 DEGREES ) '
      PRINT *,' WRIST ROLL      ( RANGE + 360 TO - 360 DEGREES ) '
      PRINT *,' WRIST PITCH     ( RANGE +  90 TO -  90 DEGREES ) '
      PRINT *,' TOOL ROLL       ( RANGE + 360 TO - 360 DEGREES ) '
      PRINT *,'   '
      PRINT *,' THE PROGRAM RETURNS THE FINAL TRANSFORMATION '
      PRINT *,' MATRIX i.e. THE POSITION & ORIENTATION MATRIX '
      PRINT *,' DEFINED AT THE WRIST PIN OR TIP OF THE ARM. '
      PRINT *,'   '
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  ENTRY OF JOINT ANGLES BY THE USER
      SUBROUTINE ANGLES(W,S,E,WR,WP,HR)
      REAL W1,S1,E1,WR1,WP1,HR1,W,S,E,WR,WP,HR,PI
C  DEFINE PI
      PI = 3.141592654
C  WAIST
 101  W = 0.0
      W1 = 0.0
      PRINT *,' ENTER WAIST ANGLE ( +/- 147 DEGREES ) ==> '
      READ(5,*) W1
      IF (ABS(W1) .GT. 147.0) THEN
        PRINT *,' ERROR -- RANGE IS +/- 147 DEGREES. '
        GOTO 101
      ENDIF
      W = W1 * PI / 180.0
C  SHOULDER
 102  S = 0.0
      S1 = 0.0
      PRINT *,' ENTER SHOULDER ANGLE ( 56 TO -236 DEGREES ) ==> '
      READ(5,*) S1
      IF ((S1 .GT. 56.0) .OR. (S1 .LT. -236.0)) THEN
        PRINT *,' ERROR -- RANGE IS 56 TO -236 DEGREES. '
        GOTO 102
      ENDIF
      S = S1 * PI / 180.0
C  ELBOW
 103  E = 0.0
      E1 = 0.0
      PRINT *,' ENTER ELBOW ANGLE ( 56 to -236 DEGREES ) ==> '
      READ(5,*) E1
      IF ((E1 .GT. 56.0) .OR. (E1 .LT. -236.0))THEN
        PRINT *,' ERROR -- RANGE IS 56 TO -236 DEGREES. '
        GOTO 103
      ENDIF
      E = E1 * PI / 180.0
C  WRIST ROLL
 104  WR = 0.0
      WR1 = 0.0
      PRINT *,' ENTER WRIST ROLL ( +/- 360 DEGREES ) ==> '
      READ(5,*) WR1
      IF (ABS(WR1) .GT. 360.0) THEN
        PRINT *,' ERROR -- RANGE IS +/- 360 DEGREES. '
        GOTO 104
      ENDIF
      WR = WR1 * PI / 180.0
C  WRIST PITCH
 105  WP = 0.0
      WP1 = 0.0
      PRINT *,' ENTER WRIST PITCH ( +/- 90 DEGREES) ==> '
      READ(5,*) WP1
      IF (ABS(WP1) .GT. 90.0) THEN
        PRINT *,' ERROR -- RANGE IS +/- 90 DEGREES. '
        GOTO 105
```

```
      ENDIF
      WP = WP1 * PI / 180.0
C  HAND ROLL
 106  HR = 0.0
      HR1 = 0.0
      PRINT *,' ENTER HAND ROLL ( +/- 360 DEGREES ) ==> '
      READ(5,*) HR1
      IF(ABS(HR1) .GT. 360.0) THEN
        PRINT *,' ERROR -- RANGE IS +/- 360 DEGREES. '
        GOTO 106
      ENDIF
      HR = HR1 * PI / 180.0
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  KINEMATICS IMPLEMENTATION
      SUBROUTINE DIRKIN(W,S,E,WR,WP,HR,T)
      INTEGER I,J,TIP
      REAL T(4,4),C1,C2,C3,C4,C5,C6,S1,S2,S3,S4,S5,S6,C23,S23,D2,D3,
     $R11A,R11B,R12A,R12B,R13A,R13B,R14A,R14B,W,S,E,WR,WP,HR,D6
C  INITIALIZE MATRIX
      DO 301 I = 1,4
        DO 301 J = 1,4
          T(I,J) = 0.0
 301  CONTINUE
      T(4,4) = 1.0
C  DEFINE COSINES AND SINES
      C1 = COS(W)
      C2 = COS(S)
      C3 = COS(E)
      C4 = COS(WR)
      C5 = COS(WP)
      C6 = COS(HR)
      S1 = SIN(W)
      S2 = SIN(S)
      S3 = SIN(E)
      S4 = SIN(WR)
      S5 = SIN(WP)
      S6 = SIN(HR)
      C23 = COS(S + E)
      S23 = SIN(S + E)
C  DEFINE D2, D3 & D6( APPROXIMATELY ) FOR THE RIGHT ARM.
C  D2 IS THE SIGNED DISTANCE FROM THE GLOBAL X AXIS TO THE SHOULDER
C  CENTRAL AXIS ( ALIGNED ALONG THE UPPER ARM ). D3 IS THE SIGNED
C  DISTANCE FROM THE SHOULDER CENTRAL AXIS TO THE ELBOW CENTRAL
C  AXIS, ALIGNED WITH THE CENTER OF THE LOWER ARM.  THESE VALUES
C  (D2 & D3) CHANGE SIGN FOR THE LEFT ARM.  D6 IS THE SIGNED
C  DISTANCE FROM THE WRIST PIN TO THE TIP OF THE ROBOT ARM.
C  ( REFER TO THE ARM KINEMATICS FOR MORE DETAILS. )
      D2 = - 19.00
      D3 = 7.00
      D6 = 3.5
C  DEFINE TRANSFORM MATRIX ENTRIES
      R11A = (C23 * ((C4 * C5 * C6)-(S4 * S6))-(S23 * S5 * C6))
```

```
      R11B = ((S4 * C5 * C6) + (C4 * S6))
      T(1,1) = (C1 * R11A) + (S1 * R11B)
      T(2,1) = (S1 * R11A) - (C1 * R11B)
      T(3,1) = -S23 * ((C4 * C5 * C6)-(S4 * S6))-(C23 * S5 * C6)
      R12A = - C23 * ((C4 * C5 * S6)+(S4 * C6))+(S23 * S5 * S6)
      R12B = ((S4 * C5 * S6)-(C4 * C6))
      T(1,2) = (C1 * R12A)-(S1 * R12B)
      T(2,2) = (S1 * R12A)+(C1 * R12B)
      T(3,2) = S23 * ((C4 * C5 * S6)+(S4 * C6))+(C23 * S5 * S6)
      R13A = (C23 * C4 * S5)+(S23 * C5)
      R13B = (S4 * S5)
      T(1,3) = (-C1 * R13A)-(S1 * R13B)
      T(2,3) = (-S1 * R13A)+(C1 * R13B)
      T(3,3) = (S23 * C4 * S5)-(C23 * C5)
      R14A = (-17.24 * S23)+(17.38 * C2)
      R14B = (D2 + D3)
      T(1,4) = (C1 * R14A)-(S1 * R14B)
      T(2,4) = (S1 * R14A)+(C1 * R14B)
      T(3,4) = (-17.24 * C23)-(17.38 * S2)
C  DECIDE ON DATA TO BE REPORTED TO TIP OR WRIST PIN
      TIP = 0
      PRINT *,' DO YOU WANT POSITION TO BE REPORTED TO TIP '
      PRINT *,' OF ROBOT ARM (1) OR WRIST PIN (0) ? '
      READ(5,*) TIP
C  IF POSITION DATA IS TO BE REPORTED W.R.T. TIP OF ROBOT ARM
C  ADD D6 * APPROACH VECTOR TO POSITION VECTOR I.E.
C  T(ROW,4) = T(ROW,4) + (T(ROW,3) * D6)
      IF(TIP .EQ. 1) THEN
        T(1,4) = T(1,4) + (T(1,3) * D6)
        T(2,4) = T(2,4) + (T(2,3) * D6)
        T(3,4) = T(3,4) + (T(3,3) * D6)
      ENDIF
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  OUTPUT TO SCREEN
      SUBROUTINE TOUT(T)
      REAL T(4,4)
      INTEGER I,J
C  OPEN DATA FILE
      OPEN(UNIT=6,STATUS='NEW',FILE='LDKIN.OUT')
      PRINT *,'   '
      DO 601 I = 1,4
C  WRITE TO SCREEN
        WRITE(5,*) (T(I,J),J=1,4)
C  WRITE TO OUTPUT FILE LDKIN.OUT
        WRITE(6,*) (T(I,J),J=1,4)
 601  CONTINUE
      PRINT *,'   '
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

APPENDIX

MERLIN 6500 MANIPULATOR INVERSE KINEMATICS SIMULATION - FORTRAN CODE

- LEFT ARM

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     MERLIN LEFT SHOULDERED ROBOT INVERSE KINEMATICS PROGRAM
C          PROGRAMMED BY :-- RANVIR S. SOLANKI
C                            WRIGHT STATE UNIVERSITY
C                            DAYTON, OH - 45435
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  MAIN PROGRAM
C  DEFINE REAL & INTEGER VALUES
       INTEGER FLAG,RESTART,WSPACE
       REAL T(4,4),Z(4,7),T1,T2P1,T2P2,T3P,T3N,T4A,T4B,T5A,T5B,
      $T6A,T6B,A2,D2,D3,D4,D6,PI,WP,WN,S1,S2,EP,EN,WR1,WR2,
      $WR3,WR4,WP1,WP2,WP3,WP4,HR1,HR2,HR3,HR4,DUM,TPP,TPN
       PRINT *,' MERLIN 6500 LEFT SHOULDER ARM '
       PRINT *,' INVERSE KINEMATICS SIMULATION '
 1     PRINT *,'    '
C  DEFINE VALUE OF CONSTANTS
       PI = 3.141592653589792
C  SETUP KINEMATIC PARAMETERS FOR THE MERLIN 6500 50 LB. ROBOT
C  A2 IS THE DISTANCE BETWEEN SHOULDER JOINT AND ELBOW JOINT
       A2 = 17.38
C  D4 IS THE DISTANCE FROM ELBOW JOINT TO WRIST PIN
       D4 = 17.24
C  D6 IS THE DISTANCE FROM WRIST PIN TO TIP OF THE END-EFFECTOR
       D6 = 3.5
C  SET UP D2 AND D3. D2 IS THE DISTANCE FROM THE WAIST VERTICAL
C  AXIS TO THE CENTER OF THE UPPER ARM. D3 IS THE DISTANCE FROM
C  THE CENTER OF THE UPPER ARM TO THE CENTER OF THE LOWER ARM.
C  FOR LEFT HAND, D2 AND D3 ARE
       D2 = 19.00
       D3 = -7.00
C  INITIALIZE ALL GLOBAL VARIABLES ( RETURNED VARIABLES ARE
C  INITIALIZED INSIDE THE SUBROUTINE ONLY )
       WP = 0.0
       S1 = 0.0
       S2 = 0.0
       EP = 0.0
       EN = 0.0
       WR1 = 0.0
       WR2 = 0.0
       WR3 = 0.0
       WR4 = 0.0
       WP1 = 0.0
       WP2 = 0.0
       WP3 = 0.0
       WP4 = 0.0
       HR1 = 0.0
       HR2 = 0.0
       HR3 = 0.0
```

```
      HR4 = 0.0
      DUM = 0.0
      T2P1 = 0.0
      T2N1 = 0.0
      T2P2 = 0.0
      T2N2 = 0.0
C  INITIALIZE [ Z ] MATRIX
C  THE FIRST COLUMN OF THE MATRIX IS A FLAG FOR VALIDITY OF THE
C  SET OF JOINT ANGLES BEING ALL WITHIN THEIR RANGES. THE REMAINING
C  4 X 6 MATRIX IS USED TO STORE THE RESULTS OF THE COMPUTATIONS
C  IN THE ORDER WAIST, SHOULDER, ELBOW, WRIST ROLL, WRIST PITCH,
C  AND HAND ROLL.
      DO 2 I = 1,4
        DO 2 J = 1,7
          Z(I,J) = 0.0
 2    CONTINUE
C  ENTER POSITION AND ORIENTATION MATRIX FROM DATAFILE OR SCREEN
 3    CALL MATENTER(T,D6)
C  FLAG SET UP FOR END POSITION IN/OUT OF WORKSPACE.
C  WSPACE = 0 IF THE END-EFFECTOR IS INSIDE THE WORKSPACE
C  WSPACE = 1 IF THE END-EFFECTOR IS OUTSIDE THE WORKSPACE
C  SET DEFAULT WSPACE FLAG = 0
      WSPACE = 0
C  COMPUTE WAIST ANGLES T1
C  IN THE CALL STATEMENT BELOW, T IS THE 4X4 POSITION AND
C  ORIENTATION WORKSPACE, T1 IS THE COMPUTED WAIST ANGLE.
      CALL WAIST(T,T1,D2,D3,WSPACE)
C  IF POSITION DESIRED AS END-POINT IS OUTSIDE THE WORKSPACE,
C  GET A NEW SET OF ENDPOINTS FROM THE USER.
      IF(WSPACE .EQ. 1) THEN
        GOTO 3
      ENDIF
C  CONVERT WAIST ANGLE FROM RADIANS TO DEGREES
C  A DUMMY VARIABLE (DUM) IS USED HERE SINCE WE ARE DEALING
C  WITH ONE WAIST ANGLE ONLY.
      CALL RADEG(T1,0.0,WP,DUM)
C  STORE RESULTS OF WAIST IN [Z] MATRIX (SECOND COLUMN)
      DO 5 I = 1,4
        Z(I,2) =  WP
 5    CONTINUE
C  RESET THE WSPACE FLAG TO 0 FOR ELBOW ANGLE COMPUTATIONS.
      WSPACE = 0
C  COMPUTE ELBOW ANGLES T3P,T3N.
      CALL ELBOW(T,T3P,T3N,A2,D2,D3,D4,WSPACE)
C  IF USER DEFINED END-POSITION IS OUTSIDE THE WORKSPACE,
C  RE-ENTRY OF MATRIX BY THE USER.
      IF(WSPACE .EQ. 1) THEN
        GOTO 3
      ENDIF
C  CONVERT ELBOW ANGLES FROM RADIANS TO DEGREES
      CALL RADEG(T3P,T3N,EP,EN)
C  STORE RESULTS OF ELBOW ANGLE SOLUTION IN THE FOURTH COLUMN
C  OF MATRIX [Z]
      DO 6 I = 1,2
```

```
        Z(I,4) = EP
        Z(I+2,4) = EN
 6    CONTINUE
C  COMPUTE ( SHOULDER + ELBOW ) ANGLES TPP,TPN
      CALL SHOULDER(T,A2,D4,T1,T3P,T3N,TPP,TPN)
C  COMPUTE SHOULDER ANGLES T2P1,T2P2
      T2P1 = TPP - T3P
      T2P2 = TPN - T3N
C  CONVERT SHOULDER ANGLES FROM RADIANS TO DEGREES
      CALL RADEG(T2P1,T2P2,S1,S2)
C  STORE RESULTS OF SHOULDER ANGLES IN [Z] MATRIX (THIRD COLUMN)
      DO 7 I = 1,2
        Z(I,3) = S1
        Z(I+2,3) = S2
 7    CONTINUE
C  COMPUTE WRIST ROLL ANGLES
      FLAG = 0
      CALL WROLL(T,T4P1,T4P2,T1,TPP,TPN)
C  CONVERT WRIST ROLL ANGLES FROM RADIANS TO DEGREES
      CALL RADEG(T4P1,T4P2,WR1,WR2)
C  COMPUTE 'WRIST FLIPPED' SOLUTIONS
      WR3 = WR1 + 180.0
      WR4 = WR2 + 180.0
C  STORE RESULTS OF WRIST ROLL IN [Z] MATRIX (FIFTH COLUMN)
      Z(1,5) = WR1
      Z(2,5) = WR3
      Z(3,5) = WR2
      Z(4,5) = WR4
C  COMPUTE WRIST PITCH ANGLES
      CALL WPITCH(T,T5P1,T5P2,T1,TPP,TPN,T4P1,T4P2)
C  CONVERT WRIST PITCH ANGLES FROM RADIANS TO DEGREES
      CALL RADEG(T5P1,T5P2,WP1,WP2)
C  COMPUTE 'REVERSED PITCH' SOLUTIONS
      WP3 = - WP1
      WP4 = - WP2
C  STORE RESULTS IN  [ Z ] MATRIX - SIXTH COLUMN
      Z(1,6) = WP1
      Z(2,6) = WP3
      Z(3,6) = WP2
      Z(4,6) = WP4
C  COMPUTE HAND ROLL
      CALL HROLL(T,T6P1,T6P2,T1,TPP,TPN,T4P1,T4P2,T5P1,T5P2)
C  CONVERT HAND ROLL ANGLES FROM RADIANS TO DEGREES
      CALL RADEG(T6P1,T6P2,HR1,HR2)
C  COMPUTE 'HAND FLIPPED' SOLUTIONS FOR HAND ROLL
      HR3 = HR1 + 180.0
      HR4 = HR2 + 180.0
C  STORE RESULTS IN THE [Z] MATRIX (SEVENTH COLUMN)
      Z(1,7) = HR1
      Z(2,7) = HR3
      Z(3,7) = HR2
      Z(4,7) = HR4
C  NORMALIZE THE COMPUTED RESULTS
      CALL NORMAL(Z)
```

```
C  CHECK FOR VALIDITY OF EACH SOLUTION SET
      CALL VALID(Z)
C  PRINT OUT VALID RESULTS ( VALID IF WITHIN JOINT ANGLE RANGE )
      PRINT *,' THE VALID INVERSE KINEMATICS RESULTS ARE :== '
      DO 51 I = 1,4
        IF(Z(I,1) .EQ. 0.0) THEN
          WRITE(5,*) 'THE VALID SOLUTION NUMBER IS ',I
          WRITE(5,*) (Z(I,J),J=2,7)
        ENDIF
 51   CONTINUE
C  QUERY FOR RESTART
      RESTART = 0
      PRINT *,' TO RESTART PROGRAM, ENTER 1 '
      PRINT *,' TO EXIT THE PROGRAM, ENTER 0 '
      READ(5,98) RESTART
      IF(RESTART .EQ. 1) THEN
        GOTO 1
      ENDIF
 98   FORMAT(I)
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  POSITION AND ORIENTATION DATA ENTRY ROUTINE
      SUBROUTINE MATENTER(A,D6)
      INTEGER MATCH,TIP,DFILE
      REAL A(4,4),PX1,PY1,PZ1,PX,PY,PZ,D6
C  INITIALIZE LOCAL VARIABLES
      PX1 = 0.0
      PY1 = 0.0
      PZ1 = 0.0
      PX  = 0.0
      PY  = 0.0
      PZ  = 0.0
C  INITIALIZE MATRIX [A]
 100  DO 101 I = 1,4
        DO 101 J = 1,4
          A(I,J) = 0.0
 101  CONTINUE
C  SET DEFAULT TO READ FROM DIRECT KINEMATICS DATA FILE
      DFILE = 1
C  READ FROM DATA FILE ?
      PRINT *,' READ TRANSFORM MATRIX FROM LDKIN.OUT ? '
      PRINT *,' ENTER 1 IF YES, 2 IF NO '
      READ(5,*) DFILE
      IF(DFILE .EQ. 1) THEN
        OPEN(UNIT=6,FILE='LDKIN.OUT',STATUS='OLD')
        DO 105 I = 1,4
          READ(6,*) (A(I,J),J=1,4)
 105    CONTINUE
        CLOSE(UNIT=6)
      ELSE
C   DATA ENTRY OF POSITION AND ORIENTATION MATRIX
        DO 102 I = 1,3
          DO 102 J = 1,4
```

```
            PRINT *,' ENTER TRANSFORM MATRIX ENTRY',I,J
            READ(5,*) A(I,J)
 102    CONTINUE
C  ADJUST ROW 4 ENTRIES TO PREVENT ENTRY ERROR
        A(4,1) = 0.0
        A(4,2) = 0.0
        A(4,3) = 0.0
        A(4,4) = 1.0
      ENDIF
C  PRINT OUT MATRIX TO SCREEN
      PRINT *,'  '
      PRINT *,' THIS IS THE ENTERED TRANSFORM MATRIX. '
      CALL AOUT(A)
      PRINT *,' IF YOU WANT TO CHANGE THE MATRIX, ENTER 0 '
      PRINT *,' IF POSITION ENTRIES REFER TO THE TIP OF THE '
      PRINT *,' END  EFFECTOR  -----  ENTER  1 '
      PRINT *,' IF POSITION ENTRIES ARE WITH RESPECT TO THE '
      PRINT *,' WRIST  PIN  ---------- ENTER 2 '
      READ(5,104) TIP
C  ALLOW FOR CHANGE OF TRANSFORM MATRIX ENTRIES
      IF(TIP .EQ. 0) THEN
        GOTO 100
      ENDIF
C  ADJUST END EFFECTOR POSITION TO WRIST PIN IF POSITION GIVEN IS
C  AT THE TIP OF THE END-EFFECTOR
      IF(TIP .EQ. 1) THEN
C  SETUP POSITION PARAMETERS TO END-EFFECTOR TIP
        PX1 = A(1,4)
        PY1 = A(2,4)
        PZ1 = A(3,4)
C  ADJUST POSITION PARAMETERS TO WRIST PIN
        PX = PX1 - D6 * A(1,3)
        PY = PY1 - D6 * A(2,3)
        PZ = PZ1 - D6 * A(3,3)
C  RESET POSITION PARAMETERS IN [A] MATRIX TO WRIST PIN
        A(1,4) = PX
        A(2,4) = PY
        A(3,4) = PZ
      ENDIF
 104  FORMAT(I)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  OUTPUT OF MATRIX TO SCREEN
      SUBROUTINE AOUT(M)
      REAL M(4,4)
      INTEGER I,J
      DO 1001 I = 1,4
        WRITE(5,*) (M(I,J),J=1,4)
 1001   CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
C  WAIST ANGLE COMPUTATION
      SUBROUTINE WAIST(A,W1,X2,X3,SPACE)
      REAL A(4,4),W1,X2,X3,RHO,PX,PY,TERM1,TERM2,
     $T2,X23
      INTEGER I,J,SPACE
C  INITIALIZATION OF LOCAL VARIABLES
      W1 = 0.0
      TERM1 = 0.0
      TERM2 = 0.0
      X23 = 0.0
      T2 = 0.0
C  SET UP OF POSITION PARAMETERS
      PX = A(1,4)
      PY = A(2,4)
C  COMPUTE FIRST TERM FOR WAIST ANGLE SOLUTION
      TERM1 = ATAN2(PY,PX)
C  COMPUTE TERM2
      X23 = (X2 + X3)
      PXSQ = PX * PX
      PYSQ = PY * PY
      PXPYSQ = PXSQ + PYSQ
      X23SQ = X23 * X23
C  USER-SPECIFIED POSITION INSIDE WORKSPACE ???
C  SET FLAG TO INSIDE WORKSPACE
      SPACE = 0
      IF(PXPYSQ .GT. X23SQ) THEN
C  SPECIFIED POSITION IS INSIDE WORK-SPACE, SO COMPUTE SECOND TERM
        GOTO 301
      ELSE
C  USER SPECIFIED POSITION IS OUTSIDE WORKSPACE.
C  COMPUTE DIFFERENCE IN TERMS
        ERROR = (ABS(PXPYSQ - X23SQ))
C  IF THE COMPUTED ERROR < 0.0001, THEN COMPUTATIONAL ERROR
C  COULD HAVE CAUSED THE POSITION TO LIE OUTSIDE THE WORKSPACE.
        IF(ERROR .LT. 0.0001) THEN
C  YES, COMPUTATIONAL ERROR OCCURED. COMPUTE T2, FOLLOWED BY
C  THE SECOND TERM.
          T2 = SQRT(ERROR)
          GOTO 302
        ELSE
C  USER SPECIFIED POSITION IS DEIFINITELY OUT OF WORKSPACE
          SPACE = 1
          PRINT *,' OUTSIDE WORKSPACE '
          GOTO 303
        ENDIF
      ENDIF
 301  T2 = SQRT(PXPYSQ - X23SQ)
 302  TERM2 = ATAN2(X23,T2)
C  COMPUTE SOLUTION FOR WAIST ANGLE W1
      W1 = TERM1 - TERM2
 303  RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
C  ELBOW ANGLE DETERMINATION ROUTINE
      SUBROUTINE ELBOW(A,EP,EN,B2,X2,X3,X4,SPACE)
      INTEGER SPACE
      REAL A(4,4),EP,EN,B2,X2,X3,X4,KA,KB,X23,T1,T2P,T2N
C  INITIALIZE LOCAL VARIABLES
      EP = 0.0
      EN = 0.0
      KA = 0.0
      KB = 0.0
      T1 = 0.0
      T2P = 0.0
      T2N = 0.0
      X23 = 0.0
C  SET UP POSITION PARAMETERS OF TRANSFORM MATRIX
      PX = A(1,4)
      PY = A(2,4)
      PZ = A(3,4)
C  COMPUTE FIRST TERM OF ARCTAN FUNCTION
      X23 = ( X2 + X3 )
      KA = - (PX * PX) - (PY * PY) - (PZ * PZ)
      KB = (B2 * B2) + (X23 * X23) + (X4 * X4)
      T1 = (KA + KB) / ( 2.0 * B2 * X4)
      T1SQ = T1 * T1
C  DETERMINE IF USER DEFINED POSITION IS OUTSIDE WORKSPACE
      SPACE = 0
C  POSITION IS INSIDE THE WORK-SPACE IF T1SQ < 1.0
      IF(T1SQ .LE. 1.0) THEN
        GOTO 401
      ELSE
C  USER DEFINED POSITION MAYBE OUTSIDE WORKSPACE
C  THEREFORE, COMPUTE THE ERROR
        ERROR = (ABS(1.0 - T1SQ))
C  CHECK TO SEE IF COMPUTATIONAL ERROR COULD HAVE CAUSED THE
C  POSITION TO LIE OUTSIDE THE WORKSPACE
        IF(ERROR .LT. 0.0001) THEN
          T2P = SQRT(ERROR)
          GOTO 402
        ELSE
C  USER ENTERED POSITION IS OUTSIDE WORKSPACE
          PRINT *,' OUTSIDE WORKSPACE '
          SPACE = 1
          GOTO 403
        ENDIF
      ENDIF
C  COMPUTE SECOND TERM OF ARCTAN FUNCTION
 401  T2P = SQRT(1.0 - T1SQ)
 402  T2N = - T2P
C  COMPUTE THE TWO POSSIBLE SOLUTIONS FOR ELBOW ANGLE I.E. EP & EN
      EP = ATAN2(T1,T2P)
      EN = ATAN2(T1,T2N)
 403  RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
C  SHOULDER + ELBOW ANGLE DETERMINATION ROUTINE
      SUBROUTINE SHOULDER(A,B2,X4,WP,EP,EN,APP,APN)
      INTEGER I,J
      REAL A(4,4),B2,X4,WP,EP,EN,T1PP,T1PN,T2PP,T2PN,C1P,S1P
     $C3P,C3N,S3P,S3N,T1PPA,T1PNA,T2PPB,T2PNB,APP,APN
C  INITIALIZE LOCAL VARIABLES
      T1PP = 0.0
      T1PN = 0.0
      T2PP = 0.0
      T2PN = 0.0
      T1PPA = 0.0
      T1PNA = 0.0
      T1PP = 0.0
      T1PN = 0.0
      T2PPB = 0.0
      T2PNB = 0.0
      APP = 0.0
      APN = 0.0
C  SETUP OF POSITION PARAMETERS
      PX = A(1,4)
      PY = A(2,4)
      PZ = A(3,4)
C  COMPUTE COSINE AND SINE FUNCTION VALUES OF THE APPROPRIATE ANGLES
      C1P = COS(WP)
      S1P = SIN(WP)
      C3P = COS(EP)
      S3P = SIN(EP)
      C3N = COS(EN)
      S3N = SIN(EN)
C  COMPUTE ALL POSSIBLE FIRST TERMS OF ARCTAN2 FUNCTION
C  WAIST POSITIVE, ELBOW POSITIVE (T1PP)
      T1PPA = B2 * C3P * PZ
      T1PP = (((B2 * S3P) - X4) * ((C1P * PX) + (S1P * PY))) - T1PPA
C  WAIST POSITIVE, ELBOW NEGATIVE (T1PN)
      T1PNA = B2 * C3N * PZ
      T1PN = (((B2 * S3N) - X4) * ((C1P * PX) + (S1P * PY))) - T1PNA
C  COMPUTE ALL POSSIBLE SECOND TERMS OF ARCTAN2 FUNCTION
C  WAIST POSITIVE, ELBOW POSITIVE (T2PP)
      T2PPB = ((B2 * C3P) * ((C1P * PX) + (S1P * PY)))
      T2PP = (((B2 * S3P) - X4) * PZ) + T2PPB
C  WAIST POSITIVE, ELBOW NEGATIVE (T2PN)
      T2PNB = ((B2 * C3N) * ((C1P * PX) + (S1P * PY)))
      T2PN = (((B2 * S3N) - X4) * PZ) + T2PNB
C  COMPUTE ALL FOUR POSSIBLE SOLUTIONS OF (THETA 2 + THETA 3)
      APP = ATAN2(T1PP,T2PP)
      APN = ATAN2(T1PN,T2PN)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  WRIST ROLL ANGLE DETERMINATION ROUTINE
      SUBROUTINE WROLL(A,PPP,PPN,WP,T23PP,T23PN)
      INTEGER FPPP,FPPN
      REAL A(4,4),PPP,PPN,WP,T23PP,T23PN,T1P,R13,R23,R33,
     $S1P,C1P,C23PP,C23PN,S23PP,S23PN,SNGCHK
```

```
C  INITIALIZE LOCAL VARIABLES
      T1P = 0.0
      T2PPP = 0.0
      T2PPN = 0.0
      PPP = 0.0
      PPN = 0.0
C  SET UP SINGULARITY CHECK CONDITION
      SNGCHK = 0.005
C  SET FLAGS TO NON-SINGULAR CASE
      FPPP = 0
      FPPN = 0
C  SETUP MATRIX ORIENTATION PARAMETERS
      R13 = A(1,3)
      R23 = A(2,3)
      R33 = A(3,3)
C  SETUP TRIG. FUNCTIONS
      S1P = SIN(WP)
      C1P = COS(WP)
      C23PP = COS(T23PP)
      S23PP = SIN(T23PP)
      C23PN = COS(T23PN)
      S23PN = SIN(T23PN)
C  COMPUTE ALL FIRST TERMS OF ARCTAN2 FUNCTION
      T1P = - (R13 * S1P) + (R23 * C1P)
C  COMPUTE ALL SECOND TERMS OF ARCTAN2 FUNCTION
      T2PPP = -(R13*C1P*C23PP) - (R23*S1P*C23PP) + (R33*S23PP)
      T2PPN = -(R13*C1P*C23PN) - (R23*S1P*C23PN) + (R33*S23PN)
C  CHECK FOR SINGULARITY CONDITIONS AT WRIST PITCH
      IF((T1P .LT. SNGCHK .AND. T1P .GT. - SNGCHK) .AND.
     $(T2PPP .LT. SNGCHK .AND. T2PPP .GT.- SNGCHK)) THEN
        FPPP = 1
      ENDIF
      IF((T1P .LT. SNGCHK .AND. T1P .GT. - SNGCHK) .AND.
     $(T2PPN .LT. SNGCHK .AND. T2PPN .GT. - SNGCHK)) THEN
        FPPN = 1
      ENDIF
C  SET WRIST ROLL TO 0.0 RADIANS IF SINGULARITY DETECTED
C  AT WRIST PITCH, ELSE COMPUTE WRIST ROLL. NOTE THAT THIS WILL
C  CAUSE THE ROLL TO SHOW UP ONLY IN HAND ROLL ANGLE.
C  SOLUTION # 1
      IF(FPPP .EQ. 1) THEN
        PPP = 0.0
      ELSE
        PPP = ATAN2(T1P,T2PPP)
      ENDIF
C  SOLUTION # 2
      IF(FPPN .EQ. 1) THEN
        PPN = 0.0
      ELSE
        PPN = ATAN2(T1P,T2PPN)
      ENDIF
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
C  WRIST PITCH DETERMINATION
      SUBROUTINE WPITCH(A,A5P1,A5P2,WP,APP,APN,W4P1,W4P2)
      REAL A(4,4),A5P1,A5P2,WP,APP,APN,W4P1,W4P2,
     $T5A1PPPP,T5A2PPPP,T5A3PPPP,T5APPPP,T5A1PPNP,T5A2PPNP,T5A3PPNP,
     $T5APPNP,T5B1PPPP,T5B2PPPP,T5B3PPPP,T5BPPPP,T5B1PPNP,T5B2PPNP,
     $T5B3PPNP,T5BPPNP,R13,R23,R33,C1P,S1P,C23PP,S23PP,
     $C23PN,S23PN,C4P1,S4P1,C4P2,S4P2
C  INITIALIZE LOCAL VARIABLES TO 0.0
      T5A1PPPP = 0.0
      T5A2PPPP = 0.0
      T5A3PPPP = 0.0
      T5APPPP  = 0.0
      T5A1PPNP = 0.0
      T5A2PPNP = 0.0
      T5A3PPNP = 0.0
      T5APPNP  = 0.0
      T5B1PPPP = 0.0
      T5B2PPPP = 0.0
      T5B3PPPP = 0.0
      T5BPPPP  = 0.0
      T5B1PPNP = 0.0
      T5B2PPNP = 0.0
      T5B3PPNP = 0.0
      T5BPPNP  = 0.0
      A5P1 = 0.0
      A5P2 = 0.0
C  SETUP ORIENTATION PARAMETERS
      R13 = A(1,3)
      R23 = A(2,3)
      R33 = A(3,3)
C  SETUP TRIG. FUNCTIONS
      C1P = COS(WP)
      S1P = SIN(WP)
      C23PP = COS(APP)
      S23PP = SIN(APP)
      C23PN = COS(APN)
      S23PN = SIN(APN)
      C4P1 = COS(W4P1)
      S4P1 = SIN(W4P1)
      C4P2 = COS(W4P2)
      S4P2 = SIN(W4P2)
C  COMPUTE FIRST TERMS OF THE ARCTAN2 FUNCTIONS
      T5A1PPPP = - (R13 * ((C1P * C23PP * C4P1) + (S1P * S4P1)))
      T5A2PPPP = - (R23 * ((S1P * C23PP * C4P1) - (C1P * S4P1)))
      T5A3PPPP = R33 * S23PP * C4P1
      T5APPPP = T5A1PPPP + T5A2PPPP + T5A3PPPP
      T5A1PPNP = - (R13 * ((C1P * C23PN * C4P2) + (S1P * S4P2)))
      T5A2PPNP = - (R23 * ((S1P * C23PN * C4P2) - (C1P * S4P2)))
      T5A3PPNP = R33 * S23PN * C4P2
      T5APPNP = T5A1PPNP + T5A2PPNP + T5A3PPNP
C  COMPUTE SECOND TERMS OF THE ARCTAN2 FUNCTIONS
      T5B1PPPP = - (C1P * S23PP * R13)
      T5B2PPPP = - (S1P * S23PP * R23)
      T5B3PPPP = - (C23PP * R33)
```

```
      T5BPPPP = T5B1PPPP + T5B2PPPP + T5B3PPPP
      T5B1PPNP = - (C1P * S23PN * R13)
      T5B2PPNP = - (S1P * S23PN * R23)
      T5B3PPNP = - (C23PN * R33)
      T5BPPNP = T5B1PPNP + T5B2PPNP + T5B3PPNP
C  COMPUTE WRIST PITCH ANGLES USING ARCTAN2 FUNCTION
      A5P1 = ATAN2(T5APPPP,T5BPPPP)
      A5P2 = ATAN2(T5APPNP,T5BPPNP)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  DETERMINATION OF HAND ROLL ANGLES
      SUBROUTINE HROLL(A,A6P1,A6P2,WP,APP,APN,A4P1,A4P2,A5P1,A5P2)
      INTEGER I,J
      REAL A(4,4),A6P1,A6P2,WP,APP,APN,A4P1,A4P2,A5P1,A5P2,
     $PPPPPA1,PPPPPA2,PPPPPA3,PPPPPA,PPNPPA1,PPNPPA2,PPNPPA3,PPNPPA
     $PPPPPB1,PPPPPB2,PPPPPB3,PPPPPB,PPNPPB1,PPNPPB2,PPNPPB3,PPNPPB
C  INITIALIZE LOCAL VARIABLES TO 0.0
      PPPPPA1 = 0.0
      PPPPPA2 = 0.0
      PPPPPA3 = 0.0
      PPPPPA  = 0.0
      PPNPPA1 = 0.0
      PPNPPA2 = 0.0
      PPNPPA3 = 0.0
      PPNPPA  = 0.0
      PPPPPB1 = 0.0
      PPPPPB2 = 0.0
      PPPPPB3 = 0.0
      PPPPPB  = 0.0
      PPNPPB1 = 0.0
      PPNPPB2 = 0.0
      PPNPPB3 = 0.0
      PPNPPB  = 0.0
C  INITIALIZE WRIST ROLL ANGLES TO 0.0
      A6P1 = 0.0
      A6P2 = 0.0
C  SETUP ROTATION PARAMETERS
      R11 = A(1,1)
      R21 = A(2,1)
      R31 = A(3,1)
C  SETUP UP TRIG. FUNCTIONS
      C1P = COS(WP)
      S1P = SIN(WP)
      C23PP = COS(APP)
      S23PP = SIN(APP)
      C23PN = COS(APN)
      S23PN = SIN(APN)
      C4P1 = COS(A4P1)
      S4P1 = SIN(A4P1)
      C4P2 = COS(A4P2)
      S4P2 = SIN(A4P2)
      C5P1 = COS(A5P1)
      S5P1 = SIN(A5P1)
```

```
      C5P2 = COS(A5P2)
      S5P2 = SIN(A5P2)
C  COMPUTE THE FIRST TERMS FOR THE ARCTAN2 FUNCTION
      PPPPPA1 = R11 * ((C1P * C23PP * S4P1) - (S1P * C4P1))
      PPPPPA2 = R21 * ((S1P * C23PP * S4P1) + (C1P * C4P1))
      PPPPPA3 = R31 * (S23PP * S4P1)
      PPPPPA = - PPPPPA1 - PPPPPA2 + PPPPPA3
      PPNPPA1 = R11 * ((C1P * C23PN * S4P2) - (S1P * C4P2))
      PPNPPA2 = R21 * ((S1P * C23PN * S4P2) + (C1P * C4P2))
      PPNPPA3 = R31 * (S23PN * S4P2)
      PPNPPA = - PPNPPA1 - PPNPPA2 + PPNPPA3
C  COMPUTE THE SECOND TERMS FOR THE ARCTAN2 FUNCTION
      PPPPPB1 = R11 * (C5P1 * ((C1P * C23PP * C4P1) + (S1P * S4P1))
     $ - (C1P * S23PP * S5P1))
      PPPPPB2 = R21 * (C5P1 * ((S1P * C23PP * C4P1) - (C1P * S4P1))
     $ - (S1P * S23PP * S5P1))
      PPPPPB3 = R31 * ((S23PP * C4P1 * C5P1) + (C23PP * S5P1))
      PPPPPB = PPPPPB1 + PPPPPB2 - PPPPPB3
      PPNPPB1 = R11 * (C5P2 * ((C1P * C23PN * C4P2) + (S1P * S4P2))
     $ - (C1P * S23PN * S5P2))
      PPNPPB2 = R21 * (C5P2 * ((S1P * C23PN * C4P2) - (C1P * S4P2))
     $ - (S1P * S23PN * S5P2))
      PPNPPB3 = R31 * ((S23PN * C4P2 * C5P2) + (C23PN * S5P2))
      PPNPPB = PPNPPB1 + PPNPPB2 - PPNPPB3
C  COMPUTE THE HAND ROLL ANGLE USING THE ARCTAN2 FUNCTION
      A6P1 = ATAN2(PPPPPA,PPPPPB)
      A6P2 = ATAN2(PPNPPA,PPNPPB)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  RADIAN TO DEGREE CONVERSION ROUTINE
      SUBROUTINE RADEG(RAD1,RAD2,DEG1,DEG2)
      REAL RAD1,RAD2,DEG1,DEG2,PI
C  INITIALIZE LOCAL VARIABLES AND RETURNED VALUES
      DEG1 = 0.0
      DEG2 = 0.0
C  DECLARE CONSTANTS
      PI = 3.141592653589792
C  PERFORM CONVERSION
      DEG1 = RAD1 * 180.0 / PI
      DEG2 = RAD2 * 180.0 / PI
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  CHECK FOR VALIDITY OF SOLUTIONS
      SUBROUTINE VALID(A)
      REAL A(4,7)
      INTEGER I,J
C  CHECK FOR VALIDITY ON ALL JOINTS.IF OUT OF RANGE,FIRST COLUMN=1.0
C  NOTE THAT THE RANGES ARE OFFSET BY 0.01 DEGREES TO TAKE CARE
C  OF COMPUTATIONAL ERRORS CAUSED BY THE MACHINE.
      DO 200 I = 1,4
C  WAIST RANGE IS FROM + 147 TO -147 DEGREES.
        IF(ABS(A(I,2) .GT. 147.01) .OR.
```

```
C  SHOULDER RANGE IS FROM + 56 TO -236 DEGREES.
     $     ((A(I,3) .GT. 56.01) .OR. (A(I,3) .LT. -236.01)) .OR.
C  ELBOW RANGE IS FROM + 56 TO -236 DEGREES.
     $     ((A(I,4) .GT. 56.01) .OR. (A(I,4) .LT. -236.01)) .OR.
C  WRIST ROLL IS CONTINUOUS. RANGE IS +/- 360 DEGREES.
     $     ABS(A(I,5) .GT. 360.01) .OR.
C  WRIST PITCH RANGE IS FROM + 90 TO -90 DEGREES.
     $     ABS(A(I,6) .GT.  90.01) .OR.
C  HAND ROLL IS CONTINUOUS. RANGE IS +/- 360 DEGREES.
     $     ABS(A(I,7) .GT. 360.01)) THEN
C  IF OUT OF RANGE, SET FLAG (COLUMN 1 OF RESPECTIVE ROW) = 1.0
         A(I,1) = 1.0
       ENDIF
 200  CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  NORMALIZE THE COMPUTED RESULTS SO THAT EACH ANGLE RANGES
C  FROM -180.0 TO 180.0 DEGREES
      SUBROUTINE NORMAL(A)
      REAL A(4,7)
      INTEGER I,J
C  NORMALIZE THE ANGLES TO BETWEEN -180 AND +180 DEGREES
      DO 701 I = 1,4
        DO 701 J = 2,7
          IF(A(I,J) .GT. 180.0) THEN
            A(I,J) = A(I,J) - 360.0
            ELSEIF (A(I,J) .LT. -180.0) THEN
              A(I,J) = A(I,J) + 360.0
          ENDIF
 701  CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

MERLIN 6500 MANIPULATOR INVERSE KINEMATICS SIMULATION - FORTRAN CODE

- RIGHT ARM

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     MERLIN ROBOT INVERSE KINEMATICS PROGRAM
C          PROGRAMMED BY :-- RANVIR S. SOLANKI
C                    WRIGHT STATE UNIVERSITY
C                    DAYTON, OH - 45435
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     MAIN PROGRAM
C     DEFINE REAL & INTEGER VALUES
      INTEGER FLAG,RESTART,WSPACE
      REAL T(4,4),Z(4,7),T1,T2P1,T2P2,T3P,T3N,T4A,T4B,T5A,T5B,
     $T6A,T6B,A2,D2,D3,D4,D6,PI,WP,WN,S1,S2,EP,EN,WR1,WR2,
     $WR3,WR4,WP1,WP2,WP3,WP4,HR1,HR2,HR3,HR4,DUM,TPP,TPN
      PRINT *,' MERLIN 6500 RIGHT SHOULDER ARM '
      PRINT *,' INVERSE KINEMATICS SIMULATION '
 1    PRINT *,'    '
C  DEFINE VALUE OF CONSTANTS
      PI = 3.141592653589792
C  SETUP KINEMATIC PARAMETERS FOR THE MERLIN 6500 50 LB. ROBOT
C  A2 IS THE DISTANCE BETWEEN SHOULDER JOINT AND ELBOW JOINT
      A2 = 17.38
C  D4 IS THE DISTANCE FROM ELBOW JOINT TO WRIST PIN
      D4 = 17.24
C  D6 IS THE DISTANCE FROM WRIST PIN TO TIP OF THE END-EFFECTOR
      D6 = 3.5
C  SET UP D2 AND D3. D2 ISTHE DISTANCE FROM THE WAIST VERTICAL
C  AXIS TO THE CENTER OF THE UPPER ARM. D3 IS THE DISTANCE FROM
C  THE CENTER OFTHE UPPER ARM TO THE CENTER OF THE LOWER ARM.
C  FOR RIGHT HAND, D2 AND D3 ARE
      D2 = -19.00
      D3 = 7.00
C  INITIALIZE ALL GLOBAL VARIABLES ( RETURNED VARIABLES ARE
C  INITIALIZED INSIDE THE SUBROUTINE ONLY )
      WP = 0.0
      S1 = 0.0
      S2 = 0.0
      EP = 0.0
      EN = 0.0
      WR1 = 0.0
      WR2 = 0.0
      WR3 = 0.0
      WR4 = 0.0
      WP1 = 0.0
      WP2 = 0.0
      WP3 = 0.0
      WP4 = 0.0
      HR1 = 0.0
      HR2 = 0.0
      HR3 = 0.0
      HR4 = 0.0
      DUM = 0.0
```

```
      T2P1 = 0.0
      T2N1 = 0.0
      T2P2 = 0.0
      T2N2 = 0.0
C  INITIALIZE [ Z ] MATRIX
C  THE FIRST COLUMN OF THE MATRIX IS A FLAG FOR VALIDITY OF THE
C  SET OF JOINT ANGLES BEING ALL WITHIN THEIR RANGES. THE REMAINING
C  4 X 6 MATRIX IS USED TO STORE THE RESULTS OF THE COMPUTATIONS
C  IN THE ORDER WAIST, SHOULDER, ELBOW, WRIST ROLL, WRIST PITCH,
C  AND HAND ROLL.
      DO 2 I = 1,4
        DO 2 J = 1,7
          Z(I,J) = 0.0
 2    CONTINUE
C  ENTER POSITION AND ORIENTATION MATRIX FROM DATAFILE OR SCREEN
 3    CALL MATENTER(T,D6)
C  FLAG SET UP FOR END POSITION IN/OUT OF WORKSPACE.
C  WSPACE = 0 IF THE END-EFFECTOR IS INSIDE THE WORKSPACE
C  WSPACE = 1 IF THE END-EFFECTOR IS OUTSIDE THE WORKSPACE
C  SET DEFAULT WSPACE FLAG = 0
      WSPACE = 0
C  COMPUTE WAIST ANGLES T1
C  IN THE CALL STATEMENT BELOW, T IS THE 4X4 POSITION AND
C  ORIENTATION WORKSPACE, T1 IS THE COMPUTED WAIST ANGLE.
      CALL WAIST(T,T1,D2,D3,WSPACE)
C  IF POSITION DESIRED AS END-POINT IS OUTSIDE THE WORKSPACE,
C  GET A NEW SET OF ENDPOINTS FROM THE USER.
      IF(WSPACE .EQ. 1) THEN
        GOTO 3
      ENDIF
C  CONVERT WAIST ANGLE FROM RADIANS TO DEGREES
C  A DUMMY VARIABLE (DUM) IS USED HERE SINCE WE ARE DEALING
C  WITH ONE WAIST ANGLE ONLY.
      CALL RADEG(T1,0.0,WP,DUM)
C  STORE RESULTS OF WAIST IN [Z] MATRIX (SECOND COLUMN)
      DO 5 I = 1,4
        Z(I,2) =  WP
 5    CONTINUE
C  RESET THE WSPACE FLAG TO 0 FOR ELBOW ANGLE COMPUTATIONS.
      WSPACE = 0
C  COMPUTE ELBOW ANGLES T3P,T3N.
      CALL ELBOW(T,T3P,T3N,A2,D2,D3,D4,WSPACE)
C  IF USER DEFINED END-POSITION IS OUTSIDE THE WORKSPACE,
C  RE-ENTRY OF MATRIX BY THE USER.
      IF(WSPACE .EQ. 1) THEN
        GOTO 3
      ENDIF
C  CONVERT ELBOW ANGLES FROM RADIANS TO DEGREES
      CALL RADEG(T3P,T3N,EP,EN)
C  STORE RESULTS OF ELBOW ANGLE SOLUTION IN THE FOURTH COLUMN
C  OF MATRIX [Z]
      DO 6 I = 1,2
        Z(I,4) = EP
        Z(I+2,4) = EN
```

```
 6    CONTINUE
C  COMPUTE ( SHOULDER + ELBOW ) ANGLES TPP,TPN
      CALL SHOULDER(T,A2,D4,T1,T3P,T3N,TPP,TPN)
C  COMPUTE SHOULDER ANGLES T2P1,T2P2
      T2P1 = TPP - T3P
      T2P2 = TPN - T3N
C  CONVERT SHOULDER ANGLES FROM RADIANS TO DEGREES
      CALL RADEG(T2P1,T2P2,S1,S2)
C  STORE RESULTS OF SHOULDER ANGLES IN [Z] MATRIX (THIRD COLUMN)
      DO 7 I = 1,2
        Z(I,3) = S1
        Z(I+2,3) = S2
 7    CONTINUE
C  COMPUTE WRIST ROLL ANGLES
      FLAG = 0
      CALL WROLL(T,T4P1,T4P2,T1,TPP,TPN)
C  CONVERT WRIST ROLL ANGLES FROM RADIANS TO DEGREES
      CALL RADEG(T4P1,T4P2,WR1,WR2)
C  COMPUTE 'WRIST FLIPPED' SOLUTIONS
      WR3 = WR1 + 180.0
      WR4 = WR2 + 180.0
C  STORE RESULTS OF WRIST ROLL IN [Z] MATRIX (FIFTH COLUMN)
      Z(1,5) = WR1
      Z(2,5) = WR3
      Z(3,5) = WR2
      Z(4,5) = WR4
C  COMPUTE WRIST PITCH ANGLES
      CALL WPITCH(T,T5P1,T5P2,T1,TPP,TPN,T4P1,T4P2)
C  CONVERT WRIST PITCH ANGLES FROM RADIANS TO DEGREES
      CALL RADEG(T5P1,T5P2,WP1,WP2)
C  COMPUTE 'REVERSED PITCH' SOLUTIONS
      WP3 = - WP1
      WP4 = - WP2
C  STORE RESULTS IN  [ Z ] MATRIX - SIXTH COLUMN
      Z(1,6) = WP1
      Z(2,6) = WP3
      Z(3,6) = WP2
      Z(4,6) = WP4
C  COMPUTE HAND ROLL
      CALL HROLL(T,T6P1,T6P2,T1,TPP,TPN,T4P1,T4P2,T5P1,T5P2)
C  CONVERT HAND ROLL ANGLES FROM RADIANS TO DEGREES
      CALL RADEG(T6P1,T6P2,HR1,HR2)
C  COMPUTE 'HAND FLIPPED' SOLUTIONS FOR HAND ROLL
      HR3 = HR1 + 180.0
      HR4 = HR2 + 180.0
C  STORE RESULTS IN THE [Z] MATRIX (SEVENTH COLUMN)
      Z(1,7) = HR1
      Z(2,7) = HR3
      Z(3,7) = HR2
      Z(4,7) = HR4
C  NORMALIZE THE COMPUTED RESULTS
      CALL NORMAL(Z)
C  CHECK FOR VALIDITY OF EACH SOLUTION SET
      CALL VALID(Z)
```

```
C  PRINT OUT VALID RESULTS ( VALID IF WITHIN JOINT ANGLE RANGE )
      PRINT *,' THE VALID INVERSE KINEMATICS RESULTS ARE :== '
      DO 51 I = 1,4
        IF(Z(I,1) .EQ. 0.0) THEN
          WRITE(5,*) 'THE VALID SOLUTION NUMBER IS ',I
          WRITE(5,*) (Z(I,J),J=2,7)
        ENDIF
 51   CONTINUE
C  QUERY FOR RESTART
      RESTART = 0
      PRINT *,' TO RESTART  PROGRAM, ENTER 1 '
      PRINT *,' TO EXIT THE PROGRAM, ENTER 0 '
      READ(5,98) RESTART
      IF(RESTART .EQ. 1) THEN
        GOTO 1
      ENDIF
 98   FORMAT(I)
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  POSITION AND ORIENTATION DATA ENTRY ROUTINE
      SUBROUTINE MATENTER(A,D6)
      INTEGER MATCH,TIP,DFILE
      REAL A(4,4),PX1,PY1,PZ1,PX,PY,PZ,D6
C  INITIALIZE LOCAL VARIABLES
      PX1 = 0.0
      PY1 = 0.0
      PZ1 = 0.0
      PX  = 0.0
      PY  = 0.0
      PZ  = 0.0
C  INITIALIZE MATRIX [A]
 100  DO 101 I = 1,4
        DO 101 J = 1,4
          A(I,J) = 0.0
 101  CONTINUE
C  SET DEFAULT TO READ FROM DIRECT KINEMATICS DATA FILE
      DFILE = 1
C  READ FROM DATA FILE ?
      PRINT *,' READ TRANSFORM MATRIX FROM LDKIN.OUT ? '
      PRINT *,' ENTER 1 IF YES, 2 IF NO '
      READ(5,*) DFILE
      IF(DFILE .EQ. 1) THEN
        OPEN(UNIT=6,FILE='LDKIN.OUT',STATUS='OLD')
        DO 105 I = 1,4
          READ(6,*) (A(I,J),J=1,4)
 105    CONTINUE
        CLOSE(UNIT=6)
      ELSE
C   DATA ENTRY OF POSITION AND ORIENTATION MATRIX
      DO 102 I = 1,3
        DO 102 J = 1,4
          PRINT *,' ENTER TRANSFORM MATRIX ENTRY',I,J
          READ(5,*) A(I,J)
```

```
 102    CONTINUE
C  ADJUST ROW 4 ENTRIES TO PREVENT ENTRY ERROR
      A(4,1) = 0.0
      A(4,2) = 0.0
      A(4,3) = 0.0
      A(4,4) = 1.0
      ENDIF
C  PRINT OUT MATRIX TO SCREEN
      PRINT *,'  '
      PRINT *,' THIS IS THE ENTERED TRANSFORM MATRIX. '
      CALL AOUT(A)
      PRINT *,' IF YOU WANT TO CHANGE THE MATRIX, ENTER 0 '
      PRINT *,' IF POSITION ENTRIES REFER TO THE TIP OF THE '
      PRINT *,' END  EFFECTOR  -----  ENTER  1 '
      PRINT *,' IF POSITION ENTRIES ARE WITH RESPECT TO THE '
      PRINT *,' WRIST  PIN  ---------- ENTER 2 '
      READ(5,104) TIP
C  ALLOW FOR CHANGE OF TRANSFORM MATRIX ENTRIES
      IF(TIP .EQ. 0) THEN
        GOTO 100
      ENDIF
C  ADJUST END EFFECTOR POSITION TO WRIST PIN IF POSITION GIVEN IS
C  AT THE TIP OF THE END-EFFECTOR
      IF(TIP .EQ. 1) THEN
C  SETUP POSITION PARAMETERS TO END-EFFECTOR TIP
        PX1 = A(1,4)
        PY1 = A(2,4)
        PZ1 = A(3,4)
C  ADJUST POSITION PARAMETERS TO WRIST PIN
        PX = PX1 - D6 * A(1,3)
        PY = PY1 - D6 * A(2,3)
        PZ = PZ1 - D6 * A(3,3)
C  RESET POSITION PARAMETERS IN [A] MATRIX TO WRIST PIN
        A(1,4) = PX
        A(2,4) = PY
        A(3,4) = PZ
      ENDIF
 104  FORMAT(I)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  OUTPUT OF MATRIX TO SCREEN
      SUBROUTINE AOUT(M)
      REAL M(4,4)
      INTEGER I,J
      DO 1001 I = 1,4
        WRITE(5,*) (M(I,J),J=1,4)
 1001   CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  WAIST ANGLE COMPUTATION
      SUBROUTINE WAIST(A,W1,X2,X3,SPACE)
      REAL A(4,4),W1,X2,X3,RHO,PX,PY,TERM1,TERM2,
```

```
     $T2,X23
      INTEGER I,J,SPACE
C  INITIALIZATION OF LOCAL VARIABLES
      W1 = 0.0
      TERM1 = 0.0
      TERM2 = 0.0
      X23 = 0.0
      T2 = 0.0
C  SET UP OF POSITION PARAMETERS
      PX = A(1,4)
      PY = A(2,4)
C  COMPUTE FIRST TERM FOR WAIST ANGLE SOLUTION
      TERM1 = ATAN2(PY,PX)
C  COMPUTE TERM2
      X23 = (X2 + X3)
      PXSQ = PX * PX
      PYSQ = PY * PY
      PXPYSQ = PXSQ + PYSQ
      X23SQ = X23 * X23
C  IS USER-SPECIFIED POSITION INSIDE THE WORKSPACE ?
C  SET FLAG TO INSIDE WORKSPACE
      SPACE = 0
      IF(PXPYSQ .GT. X23SQ) THEN
C  SPECIFIED POSITION IS INSIDE WORK-SPACE, SO COMPUTE SECOND TERM
        GOTO 301
      ELSE
C  USER SPECIFIED POSITION IS OUTSIDE WORKSPACE.
C
C  COMPUTE DIFFERENCE IN TERMS
        ERROR = (ABS(PXPYSQ - X23SQ))
C  IF THE COMPUTED ERROR < 0.0001, THEN COMPUTATIONAL ERROR
C  COULD HAVE CAUSED THE POSITION TO LIE OUTSIDE THE WORKSPACE.
        IF(ERROR .LT. 0.0001) THEN
C  YES, COMPUTATIONAL ERROR OCCURED. COMPUTE T2, FOLLOWED BY
C  THE SECOND TERM.
          T2 = SQRT(ERROR)
          GOTO 302
        ELSE
C  USER SPECIFIED POSITION IS DEFINITELY OUT OF WORKSPACE
          SPACE = 1
          PRINT *,' OUTSIDE WORKSPACE '
          GOTO 303
        ENDIF
      ENDIF
 301  T2 = SQRT(PXPYSQ - X23SQ)
 302  TERM2 = ATAN2(X23,T2)
C  COMPUTE SOLUTION FOR WAIST ANGLE W1
      W1 = TERM1 - TERM2
 303  RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  ELBOW ANGLE DETERMINATION ROUTINE
      SUBROUTINE ELBOW(A,EP,EN,B2,X2,X3,X4,SPACE)
      INTEGER SPACE
```

```
      REAL A(4,4),EP,EN,B2,X2,X3,X4,KA,KB,X23,T1,T2P,T2N
C  INITIALIZE LOCAL VARIABLES
      EP = 0.0
      EN = 0.0
      KA = 0.0
      KB = 0.0
      T1 = 0.0
      T2P = 0.0
      T2N = 0.0
      X23 = 0.0
C  SET UP POSITION PARAMETERS OF TRANSFORM MATRIX
      PX = A(1,4)
      PY = A(2,4)
      PZ = A(3,4)
C  COMPUTE FIRST TERM OF ARCTAN FUNCTION
      X23 = ( X2 + X3 )
      KA = -  (PX * PX) - (PY * PY) - (PZ * PZ)
      KB = (B2 * B2) + (X23 * X23) + (X4 * X4)
      T1 = (KA + KB) / ( 2.0 * B2 * X4)
      T1SQ = T1 * T1
C  DETERMINE IF USER DEFINED POSITION IS OUTSIDE WORKSPACE
      SPACE = 0
C  POSITION IS INSIDE THE WORK-SPACE IF T1SQ < 1.0
      IF(T1SQ .LE. 1.0) THEN
        GOTO 401
      ELSE
C  USER DEFINED POSITION MAYBE OUTSIDE WORKSPACE
C  THEREFORE, COMPUTE THE ERROR
        ERROR = (ABS(1.0 - T1SQ))
C  CHECK TO SEE IF COMPUTATIONAL ERROR COULD HAVE CAUSED THE
C  POSITION TO LIE OUTSIDE THE WORKSPACE
        IF(ERROR .LT. 0.0001) THEN
          T2P = SQRT(ERROR)
          GOTO 402
        ELSE
C  USER ENTERED POSITION IS OUTSIDE WORKSPACE
          PRINT *,' OUTSIDE WORKSPACE '
          SPACE = 1
          GOTO 403
        ENDIF
      ENDIF
C  COMPUTE SECOND TERM OF ARCTAN FUNCTION
 401  T2P = SQRT(1.0 - T1SQ)
 402  T2N = - T2P
C  COMPUTE THE TWO POSSIBLE SOLUTIONS FOR ELBOW ANGLE I.E. EP & EN
      EP = ATAN2(T1,T2P)
      EN = ATAN2(T1,T2N)
 403  RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  SHOULDER + ELBOW ANGLE DETERMINATION ROUTINE
      SUBROUTINE SHOULDER(A,B2,X4,WP,EP,EN,APP,APN)
      INTEGER I,J
      REAL A(4,4),B2,X4,WP,EP,EN,T1PP,T1PN,T2PP,T2PN,C1P,S1P
```

```
     $C3P,C3N,S3P,S3N,T1PPA,T1PNA,T2PPB,T2PNB,APP,APN
C  INITIALIZE LOCAL VARIABLES
      T1PP = 0.0
      T1PN = 0.0
      T2PP = 0.0
      T2PN = 0.0
      T1PPA = 0.0
      T1PNA = 0.0
      T1PP = 0.0
      T1PN = 0.0
      T2PPB = 0.0
      T2PNB = 0.0
      APP = 0.0
      APN = 0.0
C  SETUP OF POSITION PARAMETERS
      PX = A(1,4)
      PY = A(2,4)
      PZ = A(3,4)
C  COMPUTE COSINE AND SINE FUNCTION VALUES OF THE APPROPRIATE ANGLES
      C1P = COS(WP)
      S1P = SIN(WP)
      C3P = COS(EP)
      S3P = SIN(EP)
      C3N = COS(EN)
      S3N = SIN(EN)
C  COMPUTE ALL POSSIBLE FIRST TERMS OF ARCTAN2 FUNCTION
C
C  WAIST POSITIVE, ELBOW POSITIVE (T1PP)
      T1PPA = B2 * C3P + PZ
      T1PP = (((B2 * S3P) - X4) * ((C1P * PX) + (S1P * PY))) - T1PPA
C  WAIST POSITIVE, ELBOW NEGATIVE (T1PN)
      T1PNA = B2 * C3N * PZ
      T1PN = (((B2 * S3N) - X4) * ((C1P * PX) + (S1P * PY))) - T1PNA
C  COMPUTE ALL POSSIBLE SECOND TERMS OF ARCTAN2 FUNCTION
C
C  WAIST POSITIVE, ELBOW POSITIVE (T2PP)
      T2PPB = ((B2 * C3P) * ((C1P * PX) + (S1P * PY)))
      T2PP = (((B2 * S3P) - X4) * PZ) + T2PPB
C  WAIST POSITIVE, ELBOW NEGATIVE (T2PN)
      T2PNB = ((B2 * C3N) * ((C1P * PX) + (S1P * PY)))
      T2PN = (((B2 * S3N) - X4) * PZ) + T2PNB
C  COMPUTE ALL FOUR POSSIBLE SOLUTIONS OF (THETA 2 + THETA 3)
      APP = ATAN2(T1PP,T2PP)
      APN = ATAN2(T1PN,T2PN)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  WRIST ROLL ANGLE DETERMINATION ROUTINE
      SUBROUTINE WROLL(A,PPP,PPN,WP,T23PP,T23PN)
      INTEGER FPPP,FPPN
      REAL A(4,4),PPP,PPN,WP,T23PP,T23PN,T1P,R13,R23,R33,
     $S1P,C1P,C23PP,C23PN,S23PP,S23PN,SNGCHK
C  INITIALIZE LOCAL VARIABLES
      T1P = 0.0
```

```
      T2PPP = 0.0
      T2PPN = 0.0
      PPP = 0.0
      PPN = 0.0
C  SET UP SINGULARITY CHECK CONDITION
      SNGCHK = 0.005
C  SET FLAGS TO NON-SINGULAR CASE
      FPPP = 0
      FPPN = 0
C  SETUP MATRIX ORIENTATION PARAMETERS
      R13 = A(1,3)
      R23 = A(2,3)
      R33 = A(3,3)
C  SETUP TRIG. FUNCTIONS
      S1P = SIN(WP)
      C1P = COS(WP)
      C23PP = COS(T23PP)
      S23PP = SIN(T23PP)
      C23PN = COS(T23PN)
      S23PN = SIN(T23PN)
C  COMPUTE ALL FIRST TERMS OF ARCTAN2 FUNCTION
      T1P = - (R13 * S1P) + (R23 * C1P)
C  COMPUTE ALL SECOND TERMS OF ARCTAN2 FUNCTION
      T2PPP = -(R13*C1P*C23PP) - (R23*S1P*C23PP) + (R33*S23PP)
      T2PPN = -(R13*C1P*C23PN) - (R23*S1P*C23PN) + (R33*S23PN)
C  CHECK FOR SINGULARITY CONDITIONS AT WRIST PITCH
      IF((T1P .LT. SNGCHK .AND. T1P .GT. - SNGCHK) .AND.
     $(T2PPP .LT. SNGCHK .AND. T2PPP .GT.- SNGCHK)) THEN
          FPPP = 1
      ENDIF
      IF((T1P .LT. SNGCHK .AND. T1P .GT. - SNGCHK) .AND.
     $(T2PPN .LT. SNGCHK .AND. T2PPN .GT. - SNGCHK)) THEN
          FPPN = 1
      ENDIF
C  SET WRIST ROLL TO 0.0 RADIANS IF SINGULARITY DETECTED
C  AT WRIST PITCH, ELSE COMPUTE WRIST ROLL. NOTE THAT THIS WILL
C  CAUSE THE ROLL TO SHOW UP ONLY IN HAND ROLL ANGLE.
C  SOLUTION # 1
      IF(FPPP .EQ. 1) THEN
        PPP = 0.0
      ELSE
        PPP = ATAN2(T1P,T2PPP)
      ENDIF
C  SOLUTION # 2
      IF(FPPN .EQ. 1) THEN
        PPN = 0.0
      ELSE
        PPN = ATAN2(T1P,T2PPN)
      ENDIF
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  WRIST PITCH DETERMINATION
      SUBROUTINE WPITCH(A,A5P1,A5P2,WP,APP,APN,W4P1,W4P2)
```

```
      REAL A(4,4),A5P1,A5P2,WP,APP,APN,W4P1,W4P2,
     $T5A1PPPP,T5A2PPPP,T5A3PPPP,T5APPPP,T5A1PPNP,T5A2PPNP,T5A3PPNP,
     $T5APPNP,T5B1PPPP,T5B2PPPP,T5B3PPPP,T5BPPPP,T5B1PPNP,T5B2PPNP,
     $T5B3PPNP,T5BPPNP,R13,R23,R33,C1P,S1P,C23PP,S23PP,
     $C23PN,S23PN,C4P1,S4P1,C4P2,S4P2
C  INITIALIZE LOCAL VARIABLES TO 0.0
      T5A1PPPP = 0.0
      T5A2PPPP = 0.0
      T5A3PPPP = 0.0
      T5APPPP  = 0.0
      T5A1PPNP = 0.0
      T5A2PPNP = 0.0
      T5A3PPNP = 0.0
      T5APPNP  = 0.0
      T5B1PPPP = 0.0
      T5B2PPPP = 0.0
      T5B3PPPP = 0.0
      T5BPPPP  = 0.0
      T5B1PPNP = 0.0
      T5B2PPNP = 0.0
      T5B3PPNP = 0.0
      T5BPPNP  = 0.0
      A5P1 = 0.0
      A5P2 = 0.0
C  SETUP ORIENTATION PARAMETERS
      R13 = A(1,3)
      R23 = A(2,3)
      R33 = A(3,3)
C  SETUP TRIG. FUNCTIONS
      C1P = COS(WP)
      S1P = SIN(WP)
      C23PP = COS(APP)
      S23PP = SIN(APP)
      C23PN = COS(APN)
      S23PN = SIN(APN)
      C4P1 = COS(W4P1)
      S4P1 = SIN(W4P1)
      C4P2 = COS(W4P2)
      S4P2 = SIN(W4P2)
C  COMPUTE FIRST TERMS OF THE ARCTAN2 FUNCTIONS
      T5A1PPPP = - (R13 * ((C1P * C23PP * C4P1) + (S1P * S4P1)))
      T5A2PPPP = - (R23 * ((S1P * C23PP * C4P1) - (C1P * S4P1)))
      T5A3PPPP = R33 * S23PP * C4P1
      T5APPPP = T5A1PPPP + T5A2PPPP + T5A3PPPP
      T5A1PPNP = - (R13 * ((C1P * C23PN * C4P2) + (S1P * S4P2)))
      T5A2PPNP = - (R23 * ((S1P * C23PN * C4P2) - (C1P * S4P2)))
      T5A3PPNP = R33 * S23PN * C4P2
      T5APPNP = T5A1PPNP + T5A2PPNP + T5A3PPNP
C  COMPUTE SECOND TERMS OF THE ARCTAN2 FUNCTIONS
      T5B1PPPP = - (C1P * S23PP * R13)
      T5B2PPPP = - (S1P * S23PP * R23)
      T5B3PPPP = - (C23PP * R33)
      T5BPPPP = T5B1PPPP + T5B2PPPP + T5B3PPPP
      T5B1PPNP = - (C1P * S23PN * R13)
```

```
      T5B2PPNP = - (S1P * S23PN * R23)
      T5B3PPNP = - (C23PN * R33)
      T5BPPNP = T5B1PPNP + T5B2PPNP + T5B3PPNP
C  COMPUTE WRIST PITCH ANGLES USING ARCTAN2 FUNCTION
      A5P1 = ATAN2(T5APPPP,T5BPPPP)
      A5P2 = ATAN2(T5APPNP,T5BPPNP)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  DETERMINATION OF HAND ROLL ANGLES
      SUBROUTINE HROLL(A,A6P1,A6P2,WP,APP,APN,A4P1,A4P2,A5P1,A5P2)
      INTEGER I,J
      REAL A(4,4),A6P1,A6P2,WP,APP,APN,A4P1,A4P2,A5P1,A5P2,
     $PPPPPA1,PPPPPA2,PPPPPA3,PPPPPA,PPNPPA1,PPNPPA2,PPNPPA3,PPNPPA
     $PPPPPB1,PPPPPB2,PPPPPB3,PPPPPB,PPNPPB1,PPNPPB2,PPNPPB3,PPNPPB
C  INITIALIZE LOCAL VARIABLES TO 0.0
      PPPPPA1 = 0.0
      PPPPPA2 = 0.0
      PPPPPA3 = 0.0
      PPPPPA  = 0.0
      PPNPPA1 = 0.0
      PPNPPA2 = 0.0
      PPNPPA3 = 0.0
      PPNPPA  = 0.0
      PPPPPB1 = 0.0
      PPPPPB2 = 0.0
      PPPPPB3 = 0.0
      PPPPPB  = 0.0
      PPNPPB1 = 0.0
      PPNPPB2 = 0.0
      PPNPPB3 = 0.0
      PPNPPB  = 0.0
C  INITIALIZE WRIST ROLL ANGLES TO 0.0
      A6P1 = 0.0
      A6P2 = 0.0
C  SETUP ROTATION PARAMETERS
      R11 = A(1,1)
      R21 = A(2,1)
      R31 = A(3,1)
C  SETUP UP TRIG. FUNCTIONS
      C1P = COS(WP)
      S1P = SIN(WP)
      C23PP = COS(APP)
      S23PP = SIN(APP)
      C23PN = COS(APN)
      S23PN = SIN(APN)
      C4P1 = COS(A4P1)
      S4P1 = SIN(A4P1)
      C4P2 = COS(A4P2)
      S4P2 = SIN(A4P2)
      C5P1 = COS(A5P1)
      S5P1 = SIN(A5P1)
      C5P2 = COS(A5P2)
      S5P2 = SIN(A5P2)
```

```
C  COMPUTE THE FIRST TERMS FOR THE ARCTAN2 FUNCTION
      PPPPPA1 = R11 * ((C1P * C23PP * S4P1) - (S1P * C4P1))
      PPPPPA2 = R21 * ((S1P * C23PP * S4P1) + (C1P * C4P1))
      PPPPPA3 = R31 * (S23PP * S4P1)
      PPPPPA = - PPPPPA1 - PPPPPA2 + PPPPPA3
      PPNPPA1 = R11 * ((C1P * C23PN * S4P2) - (S1P * C4P2))
      PPNPPA2 = R21 * ((S1P * C23PN * S4P2) + (C1P * C4P2))
      PPNPPA3 = R31 * (S23PN * S4P2)
      PPNPPA = - PPNPPA1 - PPNPPA2 + PPNPPA3
C  COMPUTE THE SECOND TERMS FOR THE ARCTAN2 FUNCTION
      PPPPPB1 = R11 * (C5P1 * ((C1P * C23PP * C4P1) + (S1P * S4P1))
     $ - (C1P * S23PP * S5P1))
      PPPPPB2 = R21 * (C5P1 * ((S1P * C23PP * C4P1) - (C1P * S4P1))
     $ - (S1P * S23PP * S5P1))
      PPPPPB3 = R31 * ((S23PP * C4P1 * C5P1) + (C23PP * S5P1))
      PPPPPB = PPPPPB1 + PPPPPB2 - PPPPPB3
      PPNPPB1 = R11 * (C5P2 * ((C1P * C23PN * C4P2) + (S1P * S4P2))
     $ - (C1P * S23PN * S5P2))
      PPNPPB2 = R21 * (C5P2 * ((S1P * C23PN * C4P2) - (C1P * S4P2))
     $ - (S1P * S23PN * S5P2))
      PPNPPB3 = R31 * ((S23PN * C4P2 * C5P2) + (C23PN * S5P2))
      PPNPPB = PPNPPB1 + PPNPPB2 - PPNPPB3
C  COMPUTE THE HAND ROLL ANGLE USING THE ARCTAN2 FUNCTION
      A6P1 = ATAN2(PPPPPA,PPPPPB)
      A6P2 = ATAN2(PPNPPA,PPNPPB)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  RADIAN TO DEGREE CONVERSION ROUTINE
      SUBROUTINE RADEG(RAD1,RAD2,DEG1,DEG2)
      REAL RAD1,RAD2,DEG1,DEG2,PI
C  INITIALIZE LOCAL VARIABLES AND RETURNED VALUES
      DEG1 = 0.0
      DEG2 = 0.0
C  DECLARE CONSTANTS
      PI = 3.141592653589792
C  PERFORM CONVERSION
      DEG1 = RAD1 * 180.0 / PI
      DEG2 = RAD2 * 180.0 / PI
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  CHECK FOR VALIDITY OF SOLUTIONS
      SUBROUTINE VALID(A)
      REAL A(4,7)
      INTEGER I,J
C  CHECK FOR VALIDITY ON ALL JOINTS.IF OUT OF RANGE,FIRST COLUMN=1.0
C  NOTE THAT THE RANGES ARE OFFSET BY 0.01 DEGREES TO TAKE CARE
C  OF COMPUTATIONAL ERRORS CAUSED BY THE MACHINE.
      DO 200 I = 1,4
C  WAIST RANGE IS FROM + 147 TO -147 DEGREES.
         IF(ABS(A(I,2) .GT. 147.01) .OR.
C  SHOULDER RANGE IS FROM +56 TO -236 DEGREES.
     $     ((A(I,3) .GT. 56.01) .OR. (A(I,3) .LT. -236.01)) .OR.
```

```
C  ELBOW RANGE IS FROM + 56 TO -236 DEGREES.
     $     ((A(I,4) .GT. 56.01) .OR. (A(I,4) .LT. -236.01)) .OR.
C  WRIST ROLL IS CONTINUOUS. RANGE IS +/- 360.0 DEGREES.
     $     ABS(A(I,5) .GT. 360.01) .OR.
C  WRIST PITCH RANGE IS FROM + 90 TO - 90 DEGREES.
     $     ABS(A(I,6) .GT.  90.01) .OR.
C  HAND ROLL IS CONTINUOUS. RANGE IS +/- 360 DEGREES
     $     ABS(A(I,7) .GT. 360.01)) THEN
C  IF OUT OF RANGE,SET FLAG (COLUMN 1 OF RESPECTIVE ROW) = 1.0
          A(I,1) = 1.0
        ENDIF
 200  CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  NORMALIZE THE COMPUTED RESULTS SO THAT EACH ANGLE RANGES
C  FROM -180.0 TO 180.0 DEGREES
      SUBROUTINE NORMAL(A)
      REAL A(4,7)
      INTEGER I,J
C  NORMALIZE THE ANGLES TO BETWEEN -180 AND +180 DEGREES
      DO 701 I = 1,4
       DO 701 J = 2,7
         IF(A(I,J) .GT. 180.0) THEN
            A(I,J) = A(I,J) - 360.0
            ELSEIF(A(I,J) .LT. -180.0) THEN
            A(I,J) = A(I,J) + 360.0
            ENDIF
 701  CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

APPENDIX 5

MERLIN 6500 MANIPULATOR WORKSPACE DEVELOPMENT - FORTRAN CODE

VERTICAL WORKSPACE DEVELOPMENT

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  PROGRAM VSPACE.FOR :== MERLIN 6500 VERTICAL WORKSPACE DRAWING
C  BY :== RANVIR S. SOLANKI
C         WRIGHT STATE UNIVERSITY
C         DAYTON, OH 45435.
C  GRAPHICHS PACKAGE USED : DISSPLA
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      REAL X(1,2),Y(1,2),S,E,P,A2,A4,D6,PI
C  SET VALUE OF PI.
      PI = 3.141592653589792
C  SETUP KINEMATIC PARAMETERS FOR THE MERLIN 6500 ROBOT.
C  A2 IS THE LENGTH OF THE UPPER ARM.
      A2 = 17.38
C  A4 IS THE LENGTH OF THE LOWER ARM.
      A4 = 17.24
C  D6 IS THE DISTANCE FROM THE WRIST PIN TO THE TIP OF THE ARM.
      D6 = 3.5
C  TWO TYPES OF OUTPUT ARE ALLOWED BY THIS PROGRAM, HARDCOPY BY USING C
THE OUTPUT FILE STD00001.DAT, AND SCREEN OUTPUT ON A TEKTRONIX
C  4010 SCREEN, WHICH CAN BE SCREEN-DUMPED.
C  SETUP FOR OUTPUT FILE STD00001.DAT.  OUTPUT FILE IS CURRENTLY
C  ENABLED.
      CALL TALARS
C  SETUP FOR TEKTRONIX 4010 SCREEN OF HIGH RESOLUTION.
C  THIS OPTION IS CURRENTLY DISABLED.
C      CALL TEKALL(4010,960,0,1,0)
C  SETUP PAGE SIZE OF 8" x 8"
      CALL PAGE(8.0,8.0)
C  SETUP PLOT AREA OF 7.5" x 7.5". A BORDER OF 1/2" IS NECESSARY.
      CALL AREA2D(7.5,7.5)
C  FRAME PLOT AREA.
      CALL FRAME
C  SETUP GRAPH SCALE (X RANGE FROM -50 TO 50, Y SAME AS X)
      CALL GRAF(-50.0,'SCALE',50.0,-50.0,'SCALE',50.0)
C  WRITE TITLE ON PLOT
      CALL RLMESS('MERLIN 6500 VERTICAL WORKSPACE',30,-20.,-49.)
C
C  DRAWING OF THE LINKS AS EACH LINK MOVES DEVELOPS THE WORKSPACE
C  OF THE ROBOT ARM. THERE ARE ONLY THREE VERTICAL PLANE MOTIONS,
C  SHOULDER MOTION, ELBOW MOTION, AND WRIST PITCH MOTION.
C  DRAWING IN DISSPLA IS DONE BY CONNECTING TWO POINTS.
C  EACH POINT IS SPECIFIED BY ITS 'X' AND 'Y' POSITIONS.
C  'X' IS POSITIVE TO RIGHT OF SCREEN, 'Y' IS POSITIVE UPWARDS.
C  NO TRANSFORMATIONS ARE NEEDED, AS THE VERTICAL DIRECT KINEMATICS
C  HAVE BEEN COMPUTED WITH THE BASE FRAME SET UP ACCORDING TO THE
C  GRAPHICS FRAME, i.e., WITH X POSITIVE TO THE RIGHT, Y POSITIVE
C  UPWARDS.  THE X AND Y POSITIONS OF EACH POINT AT THE END OF THE
C  LINK ARE COMPUTED AND STORED. THUS, [X(1,1), Y(1,1)] ARE THE X,Y
```

```
C  COORDINATES OF THE FIRST POINT, WHILE [X(1,2), Y(1,2)] ARE THE
C  COORDINATES OF THE SECOND POINT.
C
C  DRAW GROUND LEVEL LINE
C
C  X AXIS POSITION OF GROUND LEVEL LINE
      X(1,1) = -45.0
      X(1,2) =  45.0
C  Y AXIS POSITION OF GROUND LEVEL LINE
      Y(1,1) = -46.45
      Y(1,2) = -46.45
C  DRAW GROUND LEVEL LINE
      CALL CURVE(X,Y,2,0)
C  INDICATE CENTER OF GROUND LEVEL LINE
      X(1,1) = 0.0
      X(1,2) = 0.0
      Y(1,1) = -45.0
      Y(1,2) = -48.0
      CALL CURVE(X,Y,2,0)
C  INDICATE GROUND LEVEL LINE ON PLOT.
      CALL RLMESS('BASE LEVEL',10,-5.0,-45.0)
C  DEVELOP VERTICAL WORKSPACE USING VERTICAL MOTION JOINT KINEMATICS.
C
C  INDICATE STEP SIZE FOR EACH JOINT ON THE PLOT.
      CALL RLMESS('SHOULDER STEP SIZE = 9.125 DEGREES',34,-48.,46.)
      CALL RLMESS('ELBOW STEP SIZE = 9.125 DEGREES',31,-48.,43.)
      CALL RLMESS('WRIST PITCH STEP SIZE = 10.0 DEGREES',36,-48.,40.)
C  DRAW UPPER ARM LINK FROM 0,0 TO END OF LINK ACCORDING TO
C  THE CURRENT SHOULDER ANGLE, IN STEPS INDICATED ABOVE.
      DO 10 S = 237.0,-57.0,-9.125
        X(1,1) = 0.0
        Y(1,1) = 0.0
        X(1,2) = A2 * COS(S * PI / 180.0)
        Y(1,2) = A2 * SIN(S * PI / 180.0)
        CALL CURVE(X,Y,2,0)
C  DRAW LOWER ARM LINK FROM BEGINNING TO END OF LINK ACCORDING TO
C  THE CURRENT ELBOW ANGLE, IN STEPS INDICATED ABOVE.
        DO 10 E = 146.0,-146.0,-9.125
          X(1,1) = A2 * COS(S * PI / 180.0)
          Y(1,1) = A2 * SIN(S * PI / 180.0)
          X(1,2) = A2 * COS(S * PI / 180.0) +
     $       A4 * COS((S + E) * PI / 180.0)
          Y(1,2) = A2 * SIN(S * PI / 180.0) +
     $       A4 * SIN((S + E) * PI / 180.0)
          CALL CURVE(X,Y,2,0)
C  DRAW WRIST PITCH LINK (WRIST PIN TO TIP OF WRIST) ACCORDING TO THE C
CURRENT WRIST PITCH ANGLE, IN STEPS INDICATED ABOVE.
          DO 10 P = 90.0,-90.0,-10.0
            X(1,1) = A2 * COS(S * PI / 180.) +
     $         A4 * COS((S + E) * PI / 180.0)
            Y(1,1) = A2 * SIN(S * PI / 180.) +
     $         A4 * SIN((S + E) * PI / 180.0)
            X(1,2) = X(1,1) + D6 * COS((S + E + P) * PI / 180.0)
            Y(1,2) = Y(1,1) + D6 * SIN((S + E + P) * PI / 180.0)
```

```
            CALL CURVE(X,Y,2,0)
 10    CONTINUE
C  CLOSE ALL DEVICES AND EXIT DISSPLA.
      CALL ENDPL(0)
      CALL DONEPL
      CALL EXIT
C  EXIT PROGRAM
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

HORIZONTAL WORKSPACE DEVELOPMENT

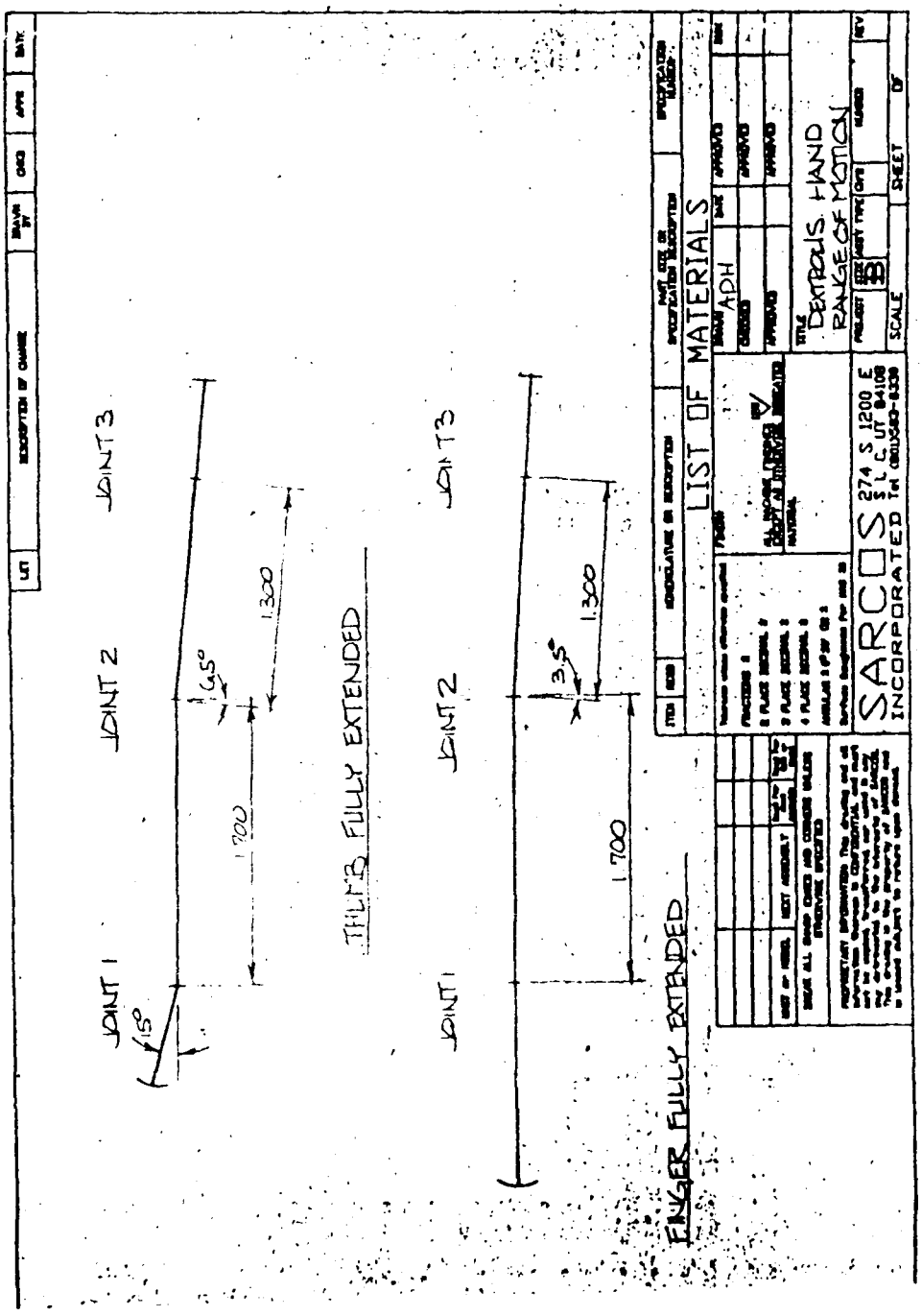
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  PROGRAM HSPACE.FOR :== MERLIN 6500 HORIZONTAL WORKSPACE DRAWING
C  BY :== RANVIR S. SOLANKI
C         WRIGHT STATE UNIVERSITY
C         DAYTON, OH 45435.
C  GRAPHICHS PACKAGE USED : DISSPLA
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      REAL X(1,2),Y(1,2),W,S,P,L,A2,A4,D,D6,PI
C  SET CONSTANT VALUES ( PI )
      PI = 3.141592653589792
C  SETUP KINEMATIC PARAMETERS FOR THE MERLIN 6500 ROBOT.
C  A2 IS THE LENGTH OF THE UPPER ARM.
      A2 = 17.38
C  A4 IS THE LENGTH OF THE LOWER ARM.
      A4 = 17.24
C  L IS THE COMBINED LENGTH OF THE UPPER AND LOWER ARMS.
      L = A2 + A4
C  D IS THE OFFSET OF THE LOWER ARM FROM THE YO AXIS.
      D = 12.0
C  D6 IS THE DISTANCE FROM THE WRIST PIN TO THE TIP OF THE ARM.
      D6 = 3.5
C  TWO TYPES OF OUTPUT ARE ALLOWED BY THIS PROGRAM, HARDCOPY BY
C  USING THE OUTPUT FILE STD00001.DAT, AND SCREEN OUTPUT ON A
C  TEKTRONIX 4010 SCREEN, WHICH CAN BE SCREEN-DUMPED.
C  SETUP FOR OUTPUT FILE STD00001.DAT.  OUTPUT FILE IS CURRENTLY
C  ENABLED.
      CALL TALARS
C  SETUP FOR TEKTRONIX 4010 SCREEN OF HIGH RESOLUTION.
C  THIS OPTION IS CURRENTLY DISABLED.
C      CALL TEKALL(4010,960,0,1,0)
C  SETUP PAGE SIZE OF 8" x 8"
      CALL PAGE(8.0,8.0)
C  SETUP PLOT AREA OF 7.5" x 7.5". A BORDER OF 1/2" IS NECESSARY.
      CALL AREA2D(7.5,7.5)
C  FRAME PLOT AREA.
      CALL FRAME
C  SETUP GRAPH SCALE (X RANGE FROM -50 TO 50, Y SAME AS X)
      CALL GRAF(-50.0,'SCALE',50.0,-50.0,'SCALE',50.0)
```

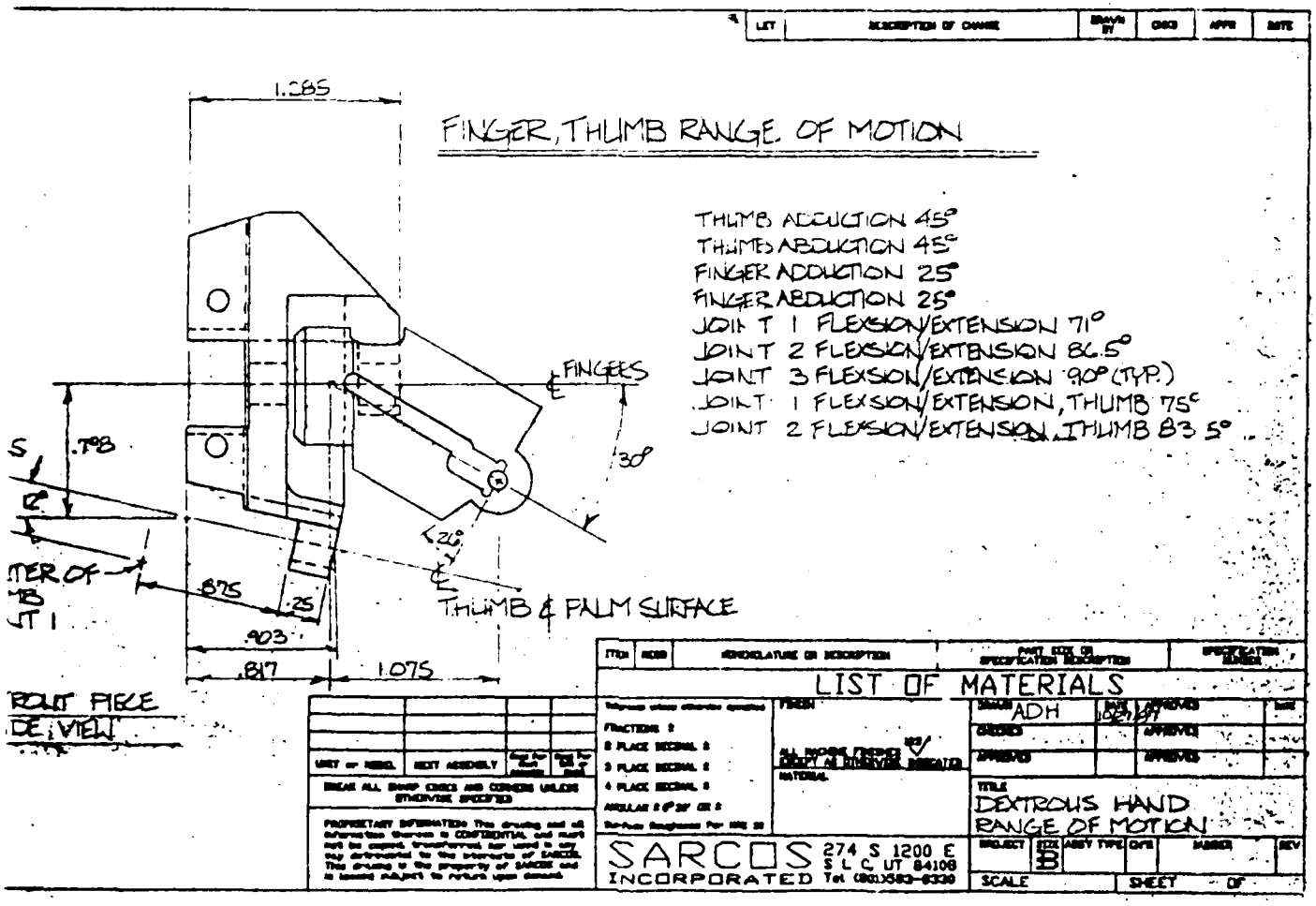
```
C  VRITE TITLE ON PLOT
      CALL RLMESS('MERLIN 6500 HORIZONTAL WORKSPACE',32,-25.,-49.)
C
C  DRAWING OF THE LINKS AS EACH LINK MOVES DEVELOPS THE WORKSPACE
C  OF THE ROBOT ARM. THERE ARE ONLY TWO HORIZONTAL PLANE MOTIONS,
C  THE WAIST MOTION AND THE WRIST YAW ( ACTUALLY THE WRIST PITCH
C  MOTION AFTER A SET ROLL OF 90 DEGREES ).
C
C  DRAWING IN DISSPLA IS DONE BY CONNECTING TWO POINTS.
C  EACH POINT IS SPECIFIED BY ITS 'X' AND 'Y' POSITIONS.
C  'X' IS POSITIVE TO RIGHT OF SCREEN, 'Y' IS POSITIVE UPWARDS.
C  NO TRANSFORMATIONS ARE NEEDED, AS THE HORIZONTAL DIRECT KINEMATICS
C  HAVE BEEN COMPUTED WITH THE BASE FRAME SET UP ACCORDING TO THE
C  GRAPHICS FRAME, i.e., WITH X POSITIVE TO THE RIGHT, Y POSITIVE
C  UPWARDS.  THE X AND Y POSITIONS OF EACH POINT AT THE END OF THE
C  LINK ARE COMPUTED AND STORED. THUS, [X(1,1), Y(1,1)] ARE THE X,Y
C  COORDINATES OF THE FIRST POINT, WHILE [X(1,2), Y(1,2)] ARE THE
C  COORDINATES OF THE SECOND POINT.
C
C  DRAW REFERENCE LINE THROUGH 0,0
C
C  X AXIS POSITION OF REFERENCE LINE.
      X(1,1) = -45.0
      X(1,2) =  45.0
C  Y AXIS POSITION OF REFERENCE LINE.
      Y(1,1) = 0.0
      Y(1,2) = 0.0
C  DRAW REFERENCE LINE.
      CALL CURVE(X,Y,2,0)
C  INDICATE CENTER OF REFERENCE LINE.
      X(1,1) = 0.0
      X(1,2) = 0.0
      Y(1,1) = 2.0
      Y(1,2) = -2.0
      CALL CURVE(X,Y,2,0)
C  INDICATE REFERENCE LINE ON PLOT.
      CALL RLMESS('REF.',4,-48.0,-3.0)
      CALL RLMESS('LINE',4,43.0,-3.0)
C  DEVELOP HORIZONTAL WORKSPACE USING HORIZONTAL JOINT KINEMATICS.
C
C  INDICATE STEP SIZE FOR EACH JOINT ON THE PLOT.
      CALL RLMESS('WAIST STEP SIZE = 3.0 DEGREES',29,-48.,46.)
      CALL RLMESS('WRIST YAW STEP SIZE = 10.0 DEGREES',34,-48.,43.)
C  DRAW SHOULDER OFFSET LINK FROM 0,0 TO END OF LINK ACCORDING TO
C  THE CURRENT WAIST ANGLE 'W', IN STEPS INDICATED ABOVE.
      DO 10 W = 147.0,-147.0,-3.0
        X(1,1) = 0.0
        Y(1,1) = 0.0
        X(1,2) = -D * COS(W * PI / 180.0)
        Y(1,2) = -D * SIN(W * PI / 180.0)
        CALL CURVE(X,Y,2,0)
C  DRAW ARM LINK FROM SHOULDER JOINT TO WRIST PIN WITH THE SHOULDER
C  ANGLE SET TO 90 DEGREES.
        X(1,1) = -D * COS(W * PI / 180.0)
```

```
      X(1,2) = -D * COS(W * PI / 180.0) +
     $   L * COS((W + 90.0) * PI / 180.0)
      Y(1,1) = -D * SIN(W * PI / 180.0)
      Y(1,2) = -D * SIN(W * PI / 180.0) +
     $   L * SIN((W + 90.0) * PI / 180.0)
      CALL CURVE(X,Y,2,0)
C  DRAW WRIST YAWING LINK (WRIST PIN TO TIP OF WRIST) ACCORDING TO
C  THE CURRENT WRIST YAW ANGLE, IN STEPS INDICATED ABOVE.
      DO 10 P = 0.0,-180.0,-10.0
        X(1,1) = -D * COS(W * PI / 180.) +
     $     L * COS((W + 90.0) * PI / 180.0)
        Y(1,1) = -D * SIN(W * PI / 180.) +
     $     L * SIN((W + 90.0) * PI / 180.0)
        X(1,2) = X(1,1) - D6 * SIN((W + 90.0 + P) * PI / 180.0)
        Y(1,2) = Y(1,1) + D6 * COS((W + 90.0 + P) * PI / 180.0)
          CALL CURVE(X,Y,2,0)
 10   CONTINUE
C  CLOSE ALL DEVICES AND EXIT DISSPLA.
      CALL ENDPL(0)
      CALL DONEPL
      CALL EXIT
C  EXIT PROGRAM
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

APPENDIX 6

MANUFACTURERS DRAWINGS OF THE FINGERS AND THUMB FOR THE UTAH/MIT DEXTEROUS HAND.




FINGER, THUMB RANGE OF MOTION
THUMB ADDUCTION 45°
THUMB ABDUCTION 45°
FINGER ADDUCTION 25°
FINGER ABDUCTION 25°
JOINT 1 FLEXION/EXTENSION 71°
JOINT 2 FLEXION/EXTENSION 86.5°
JOINT 3 FLEXION/EXTENSION 90° (TYP.)
JOINT 1 FLEXION/EXTENSION, THUMB 75°
JOINT 2 FLEXION/EXTENSION, THUMB 83.5°
1.285
.798
.875
.25
.903
.817
1.075
30°
26°
FINGERS
THUMB & PALM SURFACE
LIST OF MATERIALS
DRAWN ADH
DEXTROUS HAND
RANGE OF MOTION
SARCOS
274 S 1200 E
S L C UT 84108
INCORPORATED

APPENDIX 7

DIRECT KINEMATICS SIMULATION FOR THE UTAH/MIT DEXTEROUS HAND

PROGRAM UDKIN.FOR

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  UTAH/MIT DEXTROUS HAND DIRECT KINEMATICS SIMULATION PROGRAM
C  BY   :==   RANVIR S. SOLANKI
C             WRIGHT STATE UNIVERSITY
C             DAYTON, OHIO - 45435
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  MAIN PROGRAM
C  DEFINE REAL & INTEGER VARIABLES.
      REAL ANG(4,4),T(4,4),F1(4,4),F2(4,4),F3(4,4),PT,PF1,PF2,PF3
      INTEGER I,J,RERUN,FLAG
C  AS MULTIPLE DIGITS ARE INVOLVED IN THIS SIMULATION, WE REPRESENT
C  THE THUMB BY USING VARIABLES ENDING IN 'T', FINGER 1 VARIABLES
C  WITH 'F1', FINGER 2 WITH 'F2' AND FINGER 3 WITH 'F3'.
C  DESCRIBE PROGRAM.
      PRINT *,' THIS PROGRAM PERFORMS A MATHEMATICAL SIMULATION  '
      PRINT *,' OF THE KINEMATICS OF THE UTAH/MIT DEXTROUS HAND. '
 10   PRINT *,'  '
C  OPEN OUTPUT FILE 'HANDKIN.OUT'.
      OPEN(UNIT=6,STATUS='NEW',FILE='HANDDKIN.OUT')
C  FIND THE USER-DEFINED ANGLES FOR THE THUMB AND FINGERS.
      CALL ANGLES(ANG)
C  THE USER HAS THE OPTION OF FINDING THE POSITION W.R.T. ANY
C  POINT ON THE FINGERS OR THUMB, OR FINDING THE POSITION OF
C  THE LAST JOINT. DETERMINE IF USER WANTS POSITION W.R.T.
C  LAST JOINT ON FINGERS OR IF THE POSITION REQUIRED IS W.R.T.
C  A PARTICULAR POINT ON THE FINGER OR THUMB.
C  SET DEFAULT TO BE POSITION W.R.T. LAST JOINT.
      FLAG = 0
      CALL POSIT(FLAG,PT,PF1,PF2,PF3)
C  COMPUTE THE DIRECT KINEMATIC TRANSFORM MATRICES [T],[F1],[F2],[F3]
      CALL DIRKIN(ANG,T,F1,F2,F3,FLAG,PT,PF1,PF2,PF3)
C  OUTPUT THE DATA TO SCREEN AND A DATA FILE (HANDKIN.OUT)
      CALL TOUT(T,F1,F2,F3)
      RERUN = 0
      PRINT *,' RERUN SIMULATION ? ( 1 = YES, 0 = NO ) ==> '
      READ(5,*) RERUN
      IF(RERUN .EQ. 1) THEN
        GOTO 10
      ENDIF
C  CLOSE OUTPUT FILE
      CLOSE(UNIT=6)
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  ENTRY OF JOINT ANGLES BY THE USER
      SUBROUTINE ANGLES(ANG)
      REAL ANG(4,4),PI,ANGLE(4,4)
```

```
      INTEGER I,J,F
C  TWO MATRICES ARE DEFINED HERE, [ANG] AND [ANGLE].
C  MATRIX [ANGLE] IS THE MATRIX WHERE THE USER ENTERED VALUES FOR THE
C  JOINT ANGLES ON THE THUMB ( ROW 1 ENTRIES ) AND ON FINGERS 1,2,3.
C  ( ROW 2,3,4 ENTRIES ) ARE STORED IN DEGREES.
C  THE DEGREE VALUES OF THE ANGLES FOR THE JOINTS ON THE THUMB AND
C  FINGERS ARE THEN CONVERTED TO RADIANS AND STORED IN MATRIX [ANG].
C  JOINT 0 DATA OF EACH FINGER/THUMB IS STORED IN COLUMN 1, JOINT 1
C  DATA IN COLUMN 2, JOINT 2 DATA IN COLUMN 3 AND JOINT 3 DATA IN
C  COLUMN 4.
C  ROW 1 REPRESENTS THE THUMB VALUES, ROW 2 REPRESENTS FINGER 1
C  VALUES, ROW 3 REPRESENTS FINGER 2 VALUES,  ROW 4 REPRESENTS FINGER
C  3 VALUES IN BOTH MATRICES [ANG] AND [ANGLE]
C  DEFINE CONSTANT PI
      PI = 3.141592653589792
C  INITIALIZE MATRIX [ANG] AND [ANGLES] ENTRIES TO 0.0
      DO 100 I = 1,4
        DO 100 J = 1,4
C  USER ENTERED ANGLES MATRIX ( VALUES IN DEGREES )
          ANGLE(I,J) = 0.0
C  JOINT ANGLES MATRIX ( VALUES IN RADIANS )
          ANG(I,J) = 0.0
 100  CONTINUE
C  ENTRY BY USER OF THE THUMB ANGLES
C  JOINT 0 ON THUMB
 101  PRINT *,'ENTER THUMB JOINT 0 ANGLE (-45 TO -135 DEGREES) ==> '
      READ(5,*) ANGLE(1,1)
C  JOINT 1 ON THUMB
 102  PRINT *,'ENTER THUMB JOINT 1 ANGLE (-15 TO +60 DEGREES) ==> '
      READ(5,*) ANGLE(1,2)
C  JOINT 2 ON THUMB
 103  PRINT *,'ENTER THUMB JOINT 2 ANGLE (+6.5 TO +90 DEGREES) ==> '
      READ(5,*) ANGLE(1,3)
C  JOINT 3 ON THUMB
 104  PRINT *,'ENTER THUMB JOINT 3 ANGLE (0 TO 90 DEGREES) ==> '
      READ(5,*) ANGLE(1,4)
C  USER ENTRY OF JOINT ANGLES FOR FINGERS
      DO 110 F = 1,3
C  JOINT 0 OF FINGERS 1,2 & 3
 111  PRINT *,' ENTER FINGER ', F
      PRINT *,' JOINT 0 ANGLE ( 65 TO 115 DEGREES ) ==> '
        READ(5,*) ANGLE((F+1),1)
C  JOINT 1 OF FINGERS 1,2 & 3
 112  PRINT *,' ENTER FINGER', F
      PRINT *,' JOINT 1 ANGLE ( 120 TO 190 DEGREES ) ==>'
        READ(5,*) ANGLE((F+1),2)
C  JOINT 2 OF FINGERS 1,2 & 3
 113    PRINT *,' ENTER FINGER', F
        PRINT *,' JOINT 2 ANGLE ( 3.5 TO 90 DEGREES ) ==> '
        READ(5,*) ANGLE((F+1),3)
C  JOINT 3 OF FINGERS 1,2 & 3
 114    PRINT *,' ENTER FINGER', F
        PRINT *,' JOINT 3 ANGLE ( 0 TO 90 DEGREES ) ==> '
        READ(5,*) ANGLE((F+1),4)
```

```
 110  CONTINUE
C  CONVERT ANGLES FROM DEGREES TO RADIANS, STORE RADIAN VALUES IN
C  MATRIX [ANG]
      DO 115 I = 1,4
        DO 115 J = 1,4
          ANG(I,J) = ANGLE(I,J) * PI / 180.0
 115  CONTINUE
C  WRITE TO FILE
        WRITE(6,*) '  '
        WRITE(6,*) ' ANGLES DATA IN DEGREES. '
        WRITE(6,*) ' ROWS REPRESENT THE DIFFERENT DIGITS. '
        WRITE(6,*) ' COLUMNS REPRESENT THE JOINT NUMBERS. '
      DO 116 I = 1,4
        WRITE(6,*) (ANGLE(I,J),J=1,4)
 116  CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  USER ENTRY OF POINT ON FINGERS AND THUMB W.R.T. WHICH POSITION
C  IS TO BE REPORTED.
      SUBROUTINE POSIT(FLAG,AT,A1,A2,A3)
      REAL AT,A1,A2,A3,TMAX,F1MAX,F2MAX,F3MAX
      INTEGER FLAG
C  INITIALIZE LOCALLY COMPUTED VARIABLES TO 0.0
      AT = 0.0
      A1 = 0.0
      A2 = 0.0
      A3 = 0.0
      PRINT *,'  '
      PRINT *,' ENTER 0 IF THE POSITION DATA REPORTED IS REQUIRED '
      PRINT *,' W.R.T. THE LAST JOINT ON THE FINGERS  AND  THUMB. '
      PRINT *,' ENTER 1 IF THE POSITION DATA IS TO BE REPORTED '
      PRINT *,' W.R.T. A POINT ON THE THUMB AND FINGERS OTHER '
      PRINT *,' THAN THE LAST JOINT :  ==> '
      READ(5,300) FLAG
 300  FORMAT(I)
      IF(FLAG .EQ. 1) THEN
C  DEFINE THE MAXIMUM LENGTH OF LAST LINK OF ALL FINGERS.
        TMAX = 1.125
        F1MAX = 1.0625
        F2MAX = 1.0625
        F3MAX = 1.0625
C  DETERMINE POSITION W.R.T. WHICH DATA IS TO BE REPORTED.
 301    PRINT *,' ENTER DISTANCE ALONG THUMB ( IN INCHES ) W.R.T. '
        PRINT *,' WHICH YOU WANT THE POSITION TO BE REPORTED ==> '
        READ(5,*) AT
          IF((AT .GT. TMAX) .OR. (AT .LT. 0.0)) THEN
          PRINT *,' >>> ENTRY ERROR <<< '
          PRINT *,' MAX. LENGTH OF THUMB LAST LINK =',TMAX,'INCHES'
          GOTO 301
          ENDIF
 302    PRINT *,' ENTER DISTANCE ALONG FINGER 1 (IN INCHES) W.R.T. '
        PRINT *,' WHICH YOU WANT THE POSITION TO BE REPORTED ==> '
        READ(5,*) A1
```

```
        IF((A1 .GT. F1MAX) .OR. (A1 .LT. 0.0)) THEN
          PRINT *,' >>> ENTRY ERROR <<< '
          PRINT *,' MAX. LENGTH OF FINGER 1 LAST LINK =',F1MAX,'INCHES'
          GOTO 302
        ENDIF
 303    PRINT *,' ENTER DISTANCE ALONG FINGER 2 (IN INCHES) W.R.T. '
        PRINT *,' WHICH YOU WANT THE POSITION TO BE REPORTED ==> '
        READ(5,*) A2
        IF((A2 .GT. F2MAX) .OR. (A2 .LT. 0.0)) THEN
          PRINT *,' >>> ENTRY ERROR <<< '
          PRINT *,' MAX. LENGTH OF FINGER 2 LAST LINK =',F2MAX,'INCHES'
          GOTO 303
        ENDIF
 304    PRINT *,' ENTER DISTANCE ALONG FINGER 3 (IN INCHES) W.R.T. '
        PRINT *,' WHICH YOU WANT THE POSITION TO BE REPORTED ==> '
        READ(5,*) A3
        IF((A3 .GT. F3MAX) .OR. (A3 .LT. 0.0)) THEN
          PRINT *,' >>> ENTRY ERROR <<< '
          PRINT *,' MAX. LENGTH OF FINGER 3 LAST LINK =',F3MAX,'INCHES'
          GOTO 304
        ENDIF
      ENDIF
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  DIRECT KINEMATICS IMPLEMENTATION FOR THE UTAH/MIT HAND
      SUBROUTINE DIRKIN(ANG,T,F1,F2,F3,FLAG,AT,A1,A2,A3)
      INTEGER I,J,FLAG
      REAL ANG(4,4),T(4,4),F1(4,4),F2(4,4),F3(4,4),C1T,C2T,C23T,
     $C234T,S1T,S2T,S23T,S234T,A0T,A1T,A2T,A3T,D1T,PI,AT,A1,A2,A3,
     $A0F1,A0F2,A0F3,ANGF1(1,4),ANGF2(1,4),ANGF3(1,4),MAT(4,4)
C  MATRIX T  REFERS TO THE POSITION AND ORIENTATION MATRIX OF THUMB.
C  MATRIX F1 REFERS TO POSITION AND ORIENTATION MATRIX OF FINGER 1.
C  MATRIX F2 REFERS TO POSITION AND ORIENTATION MATRIX OF FINGER 2.
C  MATRIX F3 REFERS TO POSITION AND ORIENTATION MATRIX OF FINGER 3.
C  INITIALIZE ALL THE LOCALLY COMPUTED MATRICES
      DO 401 I = 1,4
        DO 401 J = 1,4
          T(I,J) = 0.0
          F1(I,J) = 0.0
          F2(I,J) = 0.0
          F3(I,J) = 0.0
 401  CONTINUE
C  ASSIGN THE (4,4) TERM OF ALL KINEMATIC MATRICES TO 1.0
      T(4,4) = 1.0
      F1(4,4) = 1.0
      F2(4,4) = 1.0
      F3(4,4) = 1.0
C  THUMB COMPUTATIONS
C
C  DEFINE COSINES AND SINES NEEDED FOR KINEMATICS FOR THE THUMB
      C1T = COS(ANG(1,1))
      C2T = COS(ANG(1,2))
      C23T = COS(ANG(1,2) + ANG(1,3))
```

```
      C234T = COS(ANG(1,2) + ANG(1,3) + ANG(1,4))
      S1T = SIN(ANG(1,1))
      S2T = SIN(ANG(1,2))
      S23T = SIN(ANG(1,2) + ANG(1,3))
      S234T = SIN(ANG(1,2) + ANG(1,3) + ANG(1,4))
C  DEFINE KINEMATIC PARAMETERS (A0,A1,A2,A3 & D1) FOR THUMB IN
C  INCHES.
      A0T = - 0.75
      A1T = 0.375
      A2T = 1.70
      A3T = 1.30
      D1T = 3.125
C  COMPUTE THE KINEMATICS POSITION AND ORIENTATION MATRIX FOR THUMB.
      T(1,1) = C1T * C234T
      T(1,2) = - C1T * S234T
      T(1,3) = S1T
      T(1,4) = A0T + C1T * (A1T + (A2T * C2T) + (A3T * C23T))
      T(2,1) = S1T * C234T
      T(2,2) = -S1T * S234T
      T(2,3) = -C1T
      T(2,4) = S1T * (A1T + (A2T * C2T) + (A3T * C23T))
      T(3,1) = S234T
      T(3,2) = C234T
      T(3,3) = 0.0
      T(3,4) = (A2T * S2T) + (A3T * S23T) + D1T
C  FINGER COMPUTATIONS
C  THESE TRANSFORM MATRICES ARE COMPUTED IN SUBROUTINE FINGER.
C  FINGER 1 COMPUTATIONS
C  STORE ANGLE DATA IN VECTOR ANGF1(1,4)
      DO 402 I = 1,4
        ANGF1(1,I) = ANG(2,I)
 402  CONTINUE
C  SET A0 TERM FOR FINGER 1
      A0F1 = -1.375
      CALL FINGER(ANGF1,A0F1,F1)
C  FINGER 2 COMPUTATIONS
C  STORE ANGLE DATA IN VECTOR ANGF2(1,4)
      DO 404 I = 1,4
        ANGF2(1,I) = ANG(3,I)
 404  CONTINUE
C  SET A0 TERM FOR FINGER 2
      A0F2 = 0.0
      CALL FINGER(ANGF2,A0F2,F2)
C  FINGER 3 COMPUTATIONS
C  STORE ANGLE DATA IN VECTOR ANGF3(1,4)
      DO 406 I = 1,4
        ANGF3(1,I) = ANG(4,I)
 406  CONTINUE
C  SET A0 TERM FOR FINGER 3
      A0F3 = 1.1875
      CALL FINGER(ANGF3,A0F3,F3)
C  IF THE USER HAS REQUESTED FOR POSITION TO BE REPORTED W.R.T. A
C  POINT ALONG THE LAST LINK, THEN UPDATE POSITION VECTOR TO BE
C  (POSITION VECTOR + (NORMAL VECTOR * LENGTH)).
```

```
      IF(FLAG .EQ. 1) THEN
        T(1,4) = T(1,4) + (T(1,1) * AT)
        T(2,4) = T(2,4) + (T(2,1) * AT)
        T(3,4) = T(3,4) + (T(3,1) * AT)
        F1(1,4) = F1(1,4) + (F1(1,1) * A1)
        F1(2,4) = F1(2,4) + (F1(2,1) * A1)
        F1(3,4) = F1(3,4) + (F1(3,1) * A1)
        F2(1,4) = F2(1,4) + (F2(1,1) * A2)
        F2(2,4) = F2(2,4) + (F2(2,1) * A2)
        F2(3,4) = F2(3,4) + (F2(3,1) * A2)
        F3(1,4) = F3(1,4) + (F3(1,1) * A3)
        F3(2,4) = F3(2,4) + (F3(2,1) * A3)
        F3(3,4) = F3(3,4) + (F3(3,1) * A3)
      ENDIF
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  OUTPUT TO SCREEN AND DATA FILE "HANDKIN.OUT"
      SUBROUTINE TOUT(T,F1,F2,F3)
      REAL T(4,4),F1(4,4),F2(4,4),F3(4,4)
      INTEGER I,J
C  DESCRIBE OUTPUT IN DATA FILE
      WRITE(6,*) ' TRANSFORMATION MATRICES W.R.T. BASE FRAME '
      WRITE(6,*) '  '
C  WRITE OUT THUMB MATRIX [T]
      PRINT *,' THUMB MATRIX  '
      WRITE(6,*) ' THUMB MATRIX '
      WRITE(6,*) '  '
      DO 501 I = 1,4
C  WRITE TO SCREEN
        WRITE(5,*) (T(I,J),J=1,4)
C  WRITE TO OUTPUT FILE HANDKIN.OUT
        WRITE(6,*) (T(I,J),J=1,4)
 501  CONTINUE
      PRINT *,'   '
C  WRITE OUT FINGER 1 MATRIX [F1]
      PRINT *,' FINGER 1 MATRIX  '
      WRITE(6,*) '  '
      WRITE(6,*) ' FINGER 1 MATRIX  '
      WRITE(6,*) '  '
      DO 502 I = 1,4
C  WRITE TO SCREEN
        WRITE(5,*) (F1(I,J),J=1,4)
C  WRITE TO OUTPUT FILE HANDKIN.DAT
        WRITE(6,*) (F1(I,J),J=1,4)
 502  CONTINUE
      PRINT *,'   '
C  WRITE OUT FINGER 2 MATRIX [F2]
      PRINT *,' FINGER 2 MATRIX '
      WRITE(6,*) '  '
      WRITE(6,*) ' FINGER 2 MATRIX  '
      WRITE(6,*) '  '
      DO 503 I = 1,4
C  WRITE TO SCREEN
```

```
        WRITE(5,*) (F2(I,J),J=1,4)
C  WRITE TO OUTPUT FILE HANDKIN.DAT
        WRITE(6,*) (F2(I,J),J=1,4)
 503  CONTINUE
      PRINT *,'   '
C  WRITE OUT FINGER 3 MATRIX [F3]
      PRINT *,' FINGER 3 MATRIX '
      WRITE(6,*) '  '
      WRITE(6,*) ' FINGER 3 MATRIX  '
      WRITE(6,*) '  '
      DO 504 I = 1,4
C  WRITE TO SCREEN
        WRITE(5,*) (F3(I,J),J=1,4)
C  WRITE TO OUTPUT FILE HANDKIN.DAT
        WRITE(6,*) (F3(I,J),J=1,4)
 504  CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  KINEMATIC COMPUTATIONS FOR FINGERS.
      SUBROUTINE FINGER(A,AOF,KIN)
      REAL A(1,4),KIN(4,4),AOF
      INTEGER I,J
C  INITIALIZE LOCALLY COMPUTED MATRIX [KIN]
      DO 601 I = 1,4
        DO 601 J = 1,4
          KIN(I,J) = 0.0
 601  CONTINUE
C  SET TRANSFORM MATRIX KIN[4,4] TERM TO 1.0
      KIN(4,4) = 1.0
C  DEFINE PI
      PI = 3.141592653589792
C  DEFINE THE VALUE OF ANGLE PHI IN DEGREES FOR THE FINGERS.
      PHI = 12.0
C  CONVERT ANGLE PHI TO RADIANS.
      RPHI = PHI * PI / 180.0
C  DEFINE COSINES AND SINES FOR ANGLE PHI.
      CPHI = COS(RPHI)
      SPHI = SIN(RPHI)
C  DEFINE COSINES AND SINES FOR FINGERS.
      C1F = COS(A(1,1))
      C2F = COS(A(1,2))
      C23F = COS(A(1,2) + A(1,3))
      C234F = COS(A(1,2) + A(1,3) + A(1,4))
      S1F = SIN(A(1,1))
      S2F = SIN(A(1,2))
      S23F = SIN(A(1,2) + A(1,3))
      S234F = SIN(A(1,2) + A(1,3) + A(1,4))
C  DEFINE KINEMATIC PARAMETERS (A1,A2,A3 & D1) FOR FINGERS.
      A1F = - 1.2 * SIN(30.0 * PI / 180.0)
      A2F = 1.7
      A3F = 1.3
      D1F = (4.25 / CPHI) + (1.2 * COS(30.0 * PI / 180.0))
```

```
C  COMPUTE THE KINEMATIC POSITION AND ORIENTATION MATRIX FOR FINGERS.
      KIN(1,1) = C1F * C234F
      KIN(1,2) = - C1F * S234F
      KIN(1,3) = S1F
      KIN(1,4) = A0F + C1F * (A1F + (A2F * C2F) + (A3F * C23F))
      KIN(2,1) = (SPHI * S234F) + (CPHI * S1F * C234F)
      KIN(2,2) = (SPHI * C234F) - (CPHI * S1F * S234F)
      KIN(2,3) = - CPHI * C1F
      KIN(2,4) = (D1F * SPHI) + CPHI * S1F * (A1F + (A2F * C2F)
     $+ (A3F * C23F)) + SPHI * ((A2F * S2F) + (A3F * S23F))
      KIN(3,1) = (CPHI * S234F) - (SPHI * S1F * C234F)
      KIN(3,2) = (CPHI * C234F) + (SPHI * S1F * S234F)
      KIN(3,3) = SPHI * C1F
      KIN(3,4) = (D1F * CPHI) - SPHI * S1F * (A1F + (A2F * C2F)
     $+ (A3F * C23F)) + CPHI * ((A2F * S2F) + (A3F * S23F))
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

APPENDIX 8

COMPUTER GRAPHICAL SIMULATION - FORTRAN CODE AND DOCUMENTED DATA FILES

PROGRAM SIMULATION.FOR

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DIMENSION XO(8,3,10,14),GO(3,2,10,14),TR(4,4,10,14),B(6),C(6),
     $FOCUS(2),AXDAT(8,3),XD(8,2,1000),LINKNUM(20),CF(4),M1(6,4),
     $M2(6,4),M3(6,4,100),IMITS(6,2)
      REAL XO,TR,GO,XD,D,PI,X,Y,Z,ZPOINT,B,M,ANG,FOCUS,MAG,
     $  ANGLE,TEMP,XH,YH,ZH,XP,YP,ZP,LIMITS,M1,M2,M3
      INTEGER NUM,ROBNUM,A,C,COUNT,BF,N1,N2,N,ACTIVE,
     $  AF,OPTION,SET,EXECUTE,SYSNUM,LINKNUM,RECEIVE,HANDNUM(2),
     $  ROOMNUM,MERLNUM(2),CF,END,MONITOR,INIT,KINNUM,STEPS
      CHARACTER DFILE*20
      LOGICAL DRAW(6),ERASE, REPEAT
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  SET-UP PARAMETERS
C  Z IS INITIAL VIEWPOINT IN INCHES ALONG Z AXIS
      Z=220
C  ZPOINT IS TOTAL INITIAL DISTANCE FROM DRAWING ORIGIN TO VIEWPOINT
      ZPOINT=220
C  DRAW(1X5) ARE LOGICAL VARIABLES
C  IF THE LOGICAL IS FALSE THAT UNIT WILL NOT BE DRAWN
C  IF TRUE - THAT UNIT WILL BE DRAWN
C  ROOM =              DRAW(1)
C  LEFTARM MERLIN =  DRAW(2)
C  RIGHTARM MERLIN = DRAW(3)
C  LEFTHAND UTAH =   DRAW(4)
C  RIGHTHAND UTAH =  DRAW(5)
      DRAW(1) = .TRUE.
      DRAW(2) = .TRUE.
      DRAW(3) = .FALSE.
      DRAW(4) = .TRUE.
      DRAW(5) = .FALSE.
C  MAG IS INITIAL MAGNIFICATION
      MAG=1
C  REPEAT IS LOGICAL TO TELL IF MORE THAN ONE VIEW IS TO BE DRAWN
      REPEAT = .FALSE.
C  PI VALUE OF PI
      PI=ACOS(-1.0)
C  THE ACTIVE ROBOT WILL BE THE LEFTHANDED ROBOT
      ACTIVE=1
C
C  READ ROOM.DAT
C
C    XO          STORES 3-DIMENSIONAL CORNER POINTS OF ALL LINKS
C                EXAMPLE XO(I,J,K,L)
C                I IS THE POINT NUMBER
C                J IS THE 3 DIMENSIONAL POINT VECTOR
C                K IS THE LINK NUMBER
C                L IS THE SYSTEM NUMBER
```

```
C    GO         STORES TRANSLATIONAL AND ROTATIONAL VECTORS ALL LINKS
C               GO(I,J,K,L)
C               I IS THE TRANSLATION VECTOR
C               J IS THE ROTATION VECTER
C               K IS THE LINK NUMBER
C               L IS THE SYSTEM NUMBER
C
C    LINKNUM    NUMBER OF LINKS IN EACH SYSTEM(A FINGER IS A SYSTEM)
C
C    ROBNUM           COUNTER FOR READING ROBOTS
     ROBNUM = ROBNUM + 1
C    ROOMNUM          ROOM SYSTEM NUMBERING VARIABLE
     ROOMNUM = ROBNUM
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C FILE 'ROOM.DAT' STORES THE 3-DIMENSIONAL POINTS DEFINING THE ROOM
      OPEN(UNIT=7,FILE='ROOM.DAT',STATUS = 'OLD')
      READ (7,*) LINKNUM(ROOMNUM)
      READ (7,*) (((XO(I,J,K,ROOMNUM),I=1,8),J=1,3),K=
     $  1,LINKNUM(ROBNUM))
      READ (7,*) (((GO(I,J,K,ROOMNUM),I=1,3),J=1,2),
     $    K=1,LINKNUM(ROOMNUM))
      CLOSE(UNIT=7)
C
C  READ MERL.DAT
C
      DO 87 N = 1,2
        ROBNUM=ROBNUM+1
C  MERLNUM          MERLIN ROBOT NUMBERING VARIABLE
C  MERLNUM(1) LEFT HAND , MERLNUM(2) RIGHT HAND
      MERLNUM(N) = ROBNUM
C  RECEIVE   NUMBER OF SYSTEM TO BE LINKED TO (ROBOT TO ROOM)
      RECEIVE(MERLNUM(N)) = ROOMNUM
C MERL.DAT STORES THE 3-DIMENSIONAL DATA POINTS,TRANSLATION VECTORS,
C AND ROTATION VECTORS
      OPEN(UNIT=7,FILE='MERL.DAT',STATUS = 'OLD')
      READ (7,*) LINKNUM(ROBNUM)
      READ (7,*) (((XO(I,J,K,ROBNUM),I=1,8),J=1,3),K=
     $  1,LINKNUM(ROBNUM))
      READ (7,*) (((GO(I,J,K,ROBNUM),I=1,3),J=1,2),
     $    K=1,LINKNUM(ROBNUM))
      CLOSE(UNIT=7)
  87  CONTINUE
C  ADJUST TRANSLATION MATRIX FROM FILE 'MERL.DAT' TO BUILD A
C  RIGHT HANDED ROBOT
      GO(1,1,2,MERLNUM(2))= XO(1,1,2,ROOMNUM)-GO(1,1,2,MERLNUM(2))
      GO(2,1,5,MERLNUM(2))=   -GO(2,1,5,MERLNUM(2))
      GO(3,1,6,MERLNUM(2))=   -GO(3,1,6,MERLNUM(2))
      GO(3,1,7,MERLNUM(2))=   -GO(3,1,7,MERLNUM(2))
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  UTAH HAND
C
C    HANDNUM       SYSTEM NUMBERING VARIABLE
C         HANDNUM(1) LEFT HAND   HANDNUM(2) RIGHT HAND
```

```
      DO 111 N=1,2
          HANDNUM(N)=ROBNUM + 1
          RECEIVE(HANDNUM(N)) = MERLNUM(N)
          DO 98 I=1,4
               RECEIVE(HANDNUM(N) + I) = HANDNUM(N)
  98      CONTINUE
C  'UTAH.DAT' IS DATA FILE WHICH STORES THE POINT VECTORS,TRANSLATION C
VECTORS, AND ROTATION VECTORS WITH RESPECT TO EACH LINKS OWN
C  COORDINATE SYSTEM.
          OPEN(UNIT=7,FILE='UTAH.DAT',STATUS='OLD')
          DO 109 L=1,5
               ROBNUM=ROBNUM+1
               READ(7,*)LINKNUM(ROBNUM)
                    READ(7,*)(((XO(I,J,K,ROBNUM),I=1,8),
     $                   J=1,3),K=1,LINKNUM(ROBNUM))
                    READ(7,*)(((GO(I,J,K,ROBNUM),I=1,3),
     $                   J=1,2),K=1,LINKNUM(ROBNUM))
  109          CONTINUE
          CLOSE(UNIT=7)
  111     CONTINUE
C  ADJUST TRANSLATION MATRIX FROM FILE 'UTAH.DAT' TO BUILD A
C  RIGHT HAND
      GO(1,1,2,(HANDNUM(2)+1))=-GO(1,1,2,(HANDNUM(2)+1))
      GO(1,1,2,(HANDNUM(2)+2))=-GO(1,1,2,(HANDNUM(2)+2))
      GO(1,1,2,(HANDNUM(2)+3))=-GO(1,1,2,(HANDNUM(2)+3))
      GO(1,1,2,(HANDNUM(2)+4))=-GO(1,1,2,(HANDNUM(2)+4))
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  INITIAL VIEWPOINT
C
C    INIT      INITIAL SETUP VARIABLE-CAUSES PROGRAM TO
C              CALCULATE INITIAL VIEWING ANGLES.
C
      INIT = 1
      GOTO 199
 119  INIT = 0
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  MONITOR SETUP
C
C    MONITOR        TYPE OF WORKSTATION VARIABLE
C
      WRITE(6,*)'IF WORKING ON A TEKTRONICS 4010 - ENTER 1'
      WRITE(6,*)'IF WORKING ON A TEKTRONICS 4207 - ENTER 2'
      WRITE(6,*)'IF WORKING ON A REGIS           - ENTER 3'
      READ(6,*) MONITOR
      WRITE(6,*)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  INITIAL DRAW
C  SETTING OPTION EQUAL TO ZERO TELLS THE PROGRAM WHERE TO RETURN
C  AFTER THE INITIAL DRAWING.
      OPTION = 0
C  GO TO THE DRAWING ROUTINE
      GO TO 587
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  MAIN MENU
C
C    OPTION          MENU SELECTION VARIABLE
C
 143  WRITE(6,*)'   OPTIONS'
      WRITE(6,*)'   GO TO SET-UP MENU  - ENTER 1'
      WRITE(6,*)'   GO TO EXECUTE MENU - ENTER 2'
      WRITE(6,*)'   DRAW CURRENT ROBOT - ENTER 0'
      WRITE(6,*)'   QUIT               - ENTER 10'
      READ*,OPTION
      WRITE(6,*)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  SETUP MENU
C
C    SET       SETUP MENU SELECTION VARIABLE
C
      IF (OPTION .EQ. 1) THEN
  157     WRITE(6,*)'CHANGE VIEWPOINT,           INPUT - 1'
          WRITE(6,*)'CHANGE FOCUS,AND MAGNITUDE, INPUT - 3'
          WRITE(6,*)'CHANGE SYSTEMS DRAWING,     INPUT - 4'
          WRITE(6,*)'CHANGE FIXED HAND POSITION, INPUT - 6'
          WRITE(6,*)'REPOSITION ROBOT,           INPUT - 7'
          WRITE(6,*)'CHANGE ACTIVE ROBOT         INPUT - 8'
          WRITE(6,*)'RETURN TO MAIN MENU,        INPUT - 9'
          WRITE(6,*)'DRAW ROBOT                  INPUT - 0'
          WRITE(6,*)'QUIT                        INPUT - 10'
          READ*,SET
      WRITE(6,*)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  VIEWPOINT
C
C    X,Y,Z - THE VECTOR TO THE VIEWPOINT
C
      IF (SET .EQ. 1) THEN
          WRITE(6,*)
          WRITE(6,*)'OLD VIEWPOINT - X,Y,Z'
          WRITE(6,190)X,Y,Z
  190     FORMAT('    ',3F6.0,' INCHES')
          WRITE(6,*)'INPUT - NEW VALUES - X,Y,Z IN INCHES'
          READ*,X,Y,Z
          WRITE(6,*)
C THE FOLLOWING CODE DETERMINES THE ORIENTATION OF THE ROOM AND
C EVERYTHING INSIDE.
C
C FIRST DETERMINE THE PITCH OF THE VIEWPOINT
C
  199     IF (Z .EQ. 0) THEN
            IF (X .EQ. 0) THEN
               GO(2,2,1,1)=0
               ELSE IF (X .LT. 0) THEN
                    GO(2,2,1,1)=PI/2
               ELSE
                    GO(2,2,1,1)=-PI/2
```

```
          END IF
        ELSE
             GO(2,2,1,1)=-ATAN(X/ABS(Z))
        END IF
        IF (Z .LT. 0) THEN
             GO(2,2,1,1)=PI - GO(2,2,1,1)
        END IF
C
C  ROTATE ROOM FOR A SIDE (OR OTHER) VIEW
C
        IF ((X .EQ. 0) .AND. (Z .EQ. 0)) THEN
             IF (Y .LT. 0) THEN
                  GO(1,2,1,1)=-PI/2
             ELSE
                  GO(1,2,1,1)=PI/2
             END IF
        ELSE
             GO(1,2,1,1)=ATAN(Y/SQRT((X*X)+(Z*Z)))
        END IF
C
C  DISTANCE FROM VIEWPOINT TO FOCAL POINT(WHICH IS INITIALLY [0,0,0])
C
      ZPOINT=SQRT(X*X+Y*Y+Z*Z)
      IF (INIT .EQ. 1) THEN
          GO TO 119
          ELSE IF (INIT .EQ. 2) THEN
               GO TO 502
      END IF
      WRITE(6,*)
      GO TO 157
C
C  FOCUS AND MAGNITUDE
C
C    FOCUS     TWO DIMENSIONAL SCREEN FOCAL POINT FOR VIEWING
C         X IS POSITIVE TO THE RIGHT
C         Y IS POSITIVE UPWARDS
C
C    MAG  MAGNIFICATION VALUE
C
      ELSE IF (SET .EQ. 3) THEN
      WRITE(6,*)'   OLD FOCUS X,Y IN INCHES'
      WRITE(6,259) (FOCUS(I),I=1,2)
 259  FORMAT('         ',2F6.0,' INCHES')
      WRITE(6,*)'   INPUT NEW FOCUS IN INCHES'
      READ *,(FOCUS(I),I=1,2)
      WRITE(6,*)'   OLD MAGNIFICATION'
      WRITE(6,264) MAG
 264  FORMAT('         ',F6.0)
      WRITE(6,*)'   INPUT NEW MAGNIFICATION'
      READ *,MAG
      WRITE(6,*)
      GO TO 157
C  SYTEMS DRAWING MENU
C  - CAUSES THE SYSTEMS TO BE OR NOT TO BE DRAWN
```

```
      ELSE IF (SET .EQ. 4) THEN
  270 IF (DRAW(1) .EQV. .TRUE.) THEN
          WRITE(6,*)'DO NOT DRAW ROOM,              ENTER 1'
          ELSE
          WRITE(6,*)'DRAW ROOM,                     ENTER 1'
          END IF
      IF (DRAW(2) .EQV. .TRUE.) THEN
          WRITE(6,*)'DO NOT DRAW LEFTARM MERLIN,    ENTER 2'
      ELSE
          WRITE(6,*)'DRAW LEFTARM MERLIN,           ENTER 2'
      END IF
      IF (DRAW(3) .EQV. .TRUE.) THEN
          WRITE(6,*)'DO NOT DRAW RIGHTARM MERLIN,   ENTER 3'
      ELSE
          WRITE(6,*)'DRAW RIGHTARM MERLIN,          ENTER 3'
      END IF
      IF (DRAW(4) .EQV. .TRUE.) THEN
          WRITE(6,*)'DO NOT DRAW LEFTHAND UTAH,     ENTER 4'
      ELSE
          WRITE(6,*)'DRAW LEFTHAND UTAH,            ENTER 4'
      END IF
      IF (DRAW(5) .EQV. .TRUE.) THEN
          WRITE(6,*)'DO NOT DRAW RIGHTHAND UTAH,    ENTER 5'
      ELSE
          WRITE(6,*)'DRAW RIGHTHAND UTAH,           ENTER 5'
      END IF
          WRITE(6,*)'TO QUIT,                       ENTER 6'
      READ *,N
      IF (DRAW(N) .EQV. .FALSE.) THEN
          DRAW(N) = .TRUE.
      ELSE
          DRAW(N) = .FALSE.
      END IF
      IF (N .EQ. 6) THEN
          GOTO 157
      END IF
      GOTO 270
C  POSITION HAND
C
C     ANG,ANGLE       TEMPORARY STORAGE OF ANGLES FOR READING INPUT
C
      ELSE IF (SET .EQ. 6) THEN
      WRITE(6,*)'  YOU ARE GOING TO BE ASKED FOR PITCH,
     $  YAW, AND ROLL,'
      WRITE(6,*)'  AND THEN X,Y, AND Z OFFSET FOR THE UTAH HAND'
      WRITE(6,*)'  WITH RESPECT TO THE CENTER END OF THE ARM'
C  THE YAW,PITCH, AND ROLL ANGLES ARE APPLIED TO A BLANK INITIAL
C  VECTOR MATRIX ON THE HAND, AS IS THE OFFSET VECTOR
C  YAW
C
      WRITE(6,302) GO(1,2,2,HANDNUM(ACTIVE))/PI*180
  302 FORMAT(' CURRENT YAW ANGLE IS',F8.2,' DEGREES')
      WRITE(6,*)'INPUT NEW YAW ANGLE IN DEGREES'
      READ*,ANGLE
```

```
      GO(1,2,2,HANDNUM(ACTIVE))=ANGLE/180*PI
C  PITCH
      WRITE(6,308) GO(2,2,2,HANDNUM(ACTIVE))/PI*180
  308 FORMAT(' CURRENT PITCH ANGLE IS',F8.2,' DEGREES')
      WRITE(6,*)'INPUT NEW PITCH ANGLE IN DEGREES'
      READ*,ANGLE
      GO(2,2,2,HANDNUM(ACTIVE))=ANGLE/180*PI
C  ROLL
      WRITE(6,314) GO(3,2,2,HANDNUM(ACTIVE))/PI*180
  314 FORMAT('  CURRENT ROLL ANGLE IS',F8.2,' DEGREES')
      WRITE(6,*)'INPUT NEW ROLL ANGLE IN DEGREES'
      READ*,ANGLE
      GO(3,2,2,HANDNUM(ACTIVE))=ANGLE/180*PI
C  OFFSET
C
C  XH,YH,ZH    HAND POSITIONING VARABLES
C
      WRITE(6,*)'X,Y,Z OFFSET FOR THE UTAH HAND IS MEASURED'
      WRITE(6,*)'FROM THE CENTER END OF THE ROBOT WRIST PIN'
      WRITE(6,*)'THESE VALUES ARE MEASURED WITH RESPECT TO'
      WRITE(6,*)' THE LAST COORDINATE SYSTEM OF THE MERLIN'
      WRITE(6,327)XH,YH,ZH
  327 FORMAT(' CURRENT X,Y,Z OFFSET IS ',3F8.2,' INCHES')
      WRITE(6,*)' ENTER HAND OFFSET X,Y,Z IN INCHES'
      READ *, XH,YH,ZH
      GO(1,1,2,HANDNUM(ACTIVE))=XH
      GO(2,1,2,HANDNUM(ACTIVE))=YH
      GO(3,1,2,HANDNUM(ACTIVE))=ZH
      WRITE(6,*)
      GO TO 157
C  REPOSITION ROBOT
C
C    XP,YP,ZP       ROBOT POSITIONING VALUES
C
      ELSE IF (SET .EQ. 7) THEN
      WRITE(6,*)'ENTER REPOSITION POINT OF MERLIN ROBOT'
      WRITE(6,*)'POSITIVE X IS TO THE RIGHT'
      WRITE(6,*)'POSITIVE Y IS UP'
      WRITE(6,*)'POSITIVE Z IS OUT OF THE SCREEN'
      WRITE(6,*)'THE ORIGIN IS THE LOWER,BACK,LEFT CORNER OF ROOM'
      WRITE(6,*)'THE OLD X,Y,Z POSITION IS '
      WRITE(6,347)GO(1,1,2,MERLNUM(ACTIVE)),GO(2,1,2,
     $    MERLNUM(ACTIVE)),GO(3,1,2,MERLNUM(ACTIVE))
  347 FORMAT(3F8.2,' INCHES')
      WRITE(6,*)'ENTER PLACEMENT OF BOTTOM CENTER OF ROBOT-',
     $         'X,Y,Z IN INCHES'
      READ*,XP,YP,ZP
      GO(1,1,2,MERLNUM(ACTIVE))=XP
      GO(2,1,2,MERLNUM(ACTIVE))=YP
      GO(3,1,2,MERLNUM(ACTIVE))=ZP
      WRITE(6,*)
      WRITE(6,*)'THE BASE OF THE ROBOT MAY BE ROTATED.'
      WRITE(6,*)'THE CURRENT ROTATION IN DEGREES IS'
      WRITE(6,*) GO(2,2,2,MERLNUM(ACTIVE))*180/PI,' DEGREES'
```

```
      WRITE(6,*)'ENTER BASE ROTATION IN DEGREES'
      READ*,N
      GO(2,2,2,MERLNUM(ACTIVE))=N/180*PI
      GOTO 157
C  CHANGE ACTIVE ROBOT
      ELSE IF ( SET .EQ. 8) THEN
      IF (ACTIVE .EQ. 1) THEN
          ACTIVE = 2
      ELSE
          ACTIVE = 1
      END IF
      WRITE(6,*)'ACTIVE ROBOT IS ',ACTIVE
      WRITE(6,*)
      GOTO 157
C  RETURN
      ELSE IF (SET .EQ. 9) THEN
          GOTO 143
C  REDRAW
      ELSE IF (SET .EQ. 0) THEN
          GOTO 587
C  QUIT
      ELSE IF (SET .EQ. 10) THEN
          GOTO 656
      ELSE
          GO TO 157
     END IF
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  EXECUTE MENU
C
C    EXECUTE        MENU SELECTION VARIABLE
C
      ELSE IF (OPTION .EQ. 2) THEN
  374     WRITE(6,*)'MOVE ROBOT INDIVIDUAL JOINTS,   INPUT - 1'
          WRITE(6,*)'MOVE INDIVIDUAL FINGER JOINTS,  INPUT - 2'
          WRITE(6,*)'SAVE THIS VIEW,                 INPUT - 3'
          WRITE(6,*)'DRAW ROBOT FROM SAVED DATA FILE,INPUT - 4'
          WRITE(6,*)'MOVE ROBOT TIP POINT TO POINT,  INPUT - 5'
          WRITE(6,*)'MOVE ROBOT-ALL JOINTS IN STEPS, INPUT - 6'
          WRITE(6,*)'RETURN TO MAIN MENU,            INPUT - 9'
          WRITE(6,*)'DRAW ROBOT,                     INPUT - 0'
          WRITE(6,*)'TO QUIT,                        INPUT -10'
          READ*,EXECUTE
          WRITE(6,*)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  MOVE ROBOT
C
C    C    DEFINES ACTIVE LINK NUMBERS
C
C    LIMITS    DEFINES THE LIMITS OF EACH JOINT IN DEGREES
C
      DATA C/4,5,7,8,9,10/
      DATA LIMITS/-147,-236,-236,-360,-90,-360,147,56,56,360,90,360/
      IF (EXECUTE .EQ. 1) THEN
  395     WRITE(6,*)'CHOOSE JOINT FOR CHANGE'
```

```
          WRITE(6,*) 'WAIST       -  1'
          WRITE(6,*) 'SHOULDER    -  2'
          WRITE(6,*) 'ELBOW       -  3'
          WRITE(6,*) 'WRIST ROLL  -  4'
          WRITE(6,*) 'WRIST PITCH -  5'
          WRITE(6,*) 'HAND ROLL   -  6'
          WRITE(6,*) 'TO EXIT     -  0'
          READ *, A
          IF (A .EQ. 0) THEN
               GO TO 374
          ELSE
               ANG = GO(3,2,C(A),MERLNUM(ACTIVE))*180/PI
 408           WRITE(6,409) ANG
 409           FORMAT('  OLD VALUE  ',F8.2,' DEGREES')
               WRITE(6,*)' INPUT - NEW VALUE IN DEGREES'
               READ*,D
               IF ((D .LT. LIMITS(A,1)) .OR.
    $               (D .GT. LIMITS(A,2))) THEN
                    WRITE(6,415) LIMITS(A,1),LIMITS(A,2)
 415                FORMAT(' ANGLE MUST BE BETWEEN ',
    $                    F8.2,' AND ',F8.2,' DEGREES')
                    GO TO 408
               END IF
               GO(3,2,C(A),MERLNUM(ACTIVE)) = D*PI/180
               GO TO 395
          END IF
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  MOVE FINGERS
C
C    CF IS THE SYSTEM VARIABLE
C
C    AF IS THE SYSTEM COUNTER VARIABLE
C
C    BF IS THE LINK VARIABLE
C
      ELSE IF (EXECUTE .EQ. 2) THEN
 432      WRITE(6,*)
          WRITE(6,*)'CHOOSE JOINT FOR CHANGE'
          WRITE(6,*) '0TH FINGER  -  1'
          WRITE(6,*) '1ST FINGER  -  2'
          WRITE(6,*) '2ND FINGER  -  3'
          WRITE(6,*) 'THIRD       -  4'
          WRITE(6,*) 'TO EXIT     -  0'
          READ *, AF
          WRITE(6,*)
          DO 443 I=1,4
               CF(I)=HANDNUM(ACTIVE) + I
 443           CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C    CF   FINGER JOINT VARIABLE
       IF (AF .EQ. 0) THEN
          GO TO 374
       ELSE
 450      A = CF(AF)
```

```
            WRITE(6,*)'CHOOSE POSITION FOR CHANGE'
            WRITE(6,*) '1ST JOINT   -  1'
            WRITE(6,*) '2ND JOINT   -  2'
            WRITE(6,*) '3RD JOINT   -  3'
            WRITE(6,*) '4TH JOINT   -  4'
            WRITE(6,*) 'TO EXIT     -  0'
            READ *, BF
         END IF
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
         IF (BF .EQ. 0) THEN
            GO TO 432
         ELSE
            BF = BF+1
            WRITE(6,*)
            WRITE(6,466)(GO(3,2,BF,A))*180/PI
  466       FORMAT('   OLD VALUE  ',F8.2,' DEGREES')
            WRITE(6,*)'  INPUT - NEW VALUE IN DEGREES'
            READ*,D
            GO(3,2,BF,A) = D*PI/180
            WRITE(6,*)
            GO TO 450
         END IF
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  SAVE VIEW
C STORES THE VARIABLE LINK PARAMETERS IN AN OUTPUT FILE
C
       ELSE IF (EXECUTE .EQ. 3) THEN
       OPEN(UNIT=8,FILE='SAVE.DAT',STATUS='NEW')
       WRITE(8,*) X,Y,Z,ZPOINT,MAG,FOCUS
       DO 484 I=1,ROBNUM
           DO 484 J=1,LINKNUM(ROBNUM)
                WRITE(8,*)(GO(L,1,J,I),L=1,3)
                WRITE(8,*)(GO(L,2,J,I),L=1,3)
  484  CONTINUE
       CLOSE(UNIT=8)
       GO TO 374
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
       ELSE IF (EXECUTE .EQ. 4) THEN
C  DRAW FROM THE LAST SAVED VIEW
C  CALLS THE VARIABLE LINK VALUES FROM AN OUTFILE AND DRAWS THE ROBOT
       OPEN(UNIT=8,FILE='SAVE.DAT',STATUS='OLD')
       READ(8,*) X,Y,Z,ZPOINT,MAG,FOCUS
       DO 498 I=1,ROBNUM
           DO 498 J=1,LINKNUM(ROBNUM)
                READ(8,*)(GO(L,1,J,I),L=1,3)
                READ(8,*)(GO(L,2,J,I),L=1,3)
  498  CONTINUE
       INIT = 2
       GO TO 199
  502  INIT = 0
       CLOSE(UNIT=8)
       GO TO 587
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
       ELSE IF (EXECUTE .EQ. 5) THEN
```

```
C  MOVE TO POINT
C
C    B     STORAGE OF JOINT ANGLES
C
C    KINNUM    PARAMETER PASSED IN CALL INKIN
C    KINNUM=1 (LEFT HAND),KINNUM=2 (RIGHT HAND)
C
C  INKIN PROVIDES THE ANGLES TO DRAW TO A CERTAIN POINT
      CALL INKIN(B,KINNUM)
      DO 518 I=1,6
          GO(3,2,C(I),MERLNUM(ACTIVE))=B(I)/180*PI
  518 CONTINUE
      GO TO 587
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      ELSE IF (EXECUTE .EQ. 6) THEN
C  MOVE BY ANGLES
C
C    NUM       NUMBER READING VARIABLE
C
C    STEPS     NUMBER OF INCREMENTS DURING MOTION
C
C    ERASE     LOGICAL - IS SCREEN ERASED BETWEEN VIEWS
C
C    REPEAT    LOGICAL FOR INTERNAL MEMORY CONCERNING DRAWING SETUP
C
  533 WRITE(6,*)'INPUT THE SIX ANGLES,
     $ WAIST,SHOULDER,ELBOW,WRISTROLL,WRISTPITCH,HANDROLL)'
      WRITE(6,*)'CURRENT ANGLES IN DEGREES ARE'
      WRITE(6,36)(GO(3,2,C(I),MERLNUM(ACTIVE))*180/PI,I=1,6)
  36  FORMAT(6F7.1)
      READ(6,*)(M2(I,1),I=1,6)
      WRITE(6,*)'DO YOU WANT THE SCREEN ERASED BETWEEN STEPS'
      WRITE(6,*)'IF YES INPUT    1'
      WRITE(6,*)'IF NO  INPUT    0'
      READ*,NUM
      IF (NUM .EQ. 1) THEN
          ERASE = .TRUE.
          ELSE
               ERASE = .FALSE.
      END IF
      DO 556 I=1,6
          IF ((M2(I,1) .LT. LIMITS(I,1)) .OR.
     $         (M2(I,1) .GT. LIMITS(I,2))) THEN
               WRITE(6,552) I,LIMITS(I,1),LIMITS(I,2)
  552          FORMAT('ANGLE- ',I2,' MUST BE BETWEEN ',
     $              F7.1,' AND ',F7.1,' DEGREES')
               GO TO 533
          END IF
  556 CONTINUE
      DO 560 I=1,6
          M2(I,1)=M2(I,1)/180*PI
          M1(I,1)=GO(3,2,C(I),MERLNUM(ACTIVE))
  560 CONTINUE
```

```
C  M1 IS THE CURRENT SET OF ANGLES, M2 IS THE FINAL SET(TO BE INPUT
C  BY USER), M3 IS RETURNED FROM GENANG AND CONTAINS THE EQUALLY
C  SPACED INCREMENTAL SET  OF STEP NUMBER OF ANGLES.
C
      CALL GENANG(M1,M2,6,1,M3,STEPS)
      DO 573 J=1,STEPS
          IF (J .EQ. STEPS) THEN
            REPEAT = .FALSE.
            ELSE
            REPEAT = .TRUE.
          END IF
          DO 570 I=1,6
              GO(3,2,C(I),MERLNUM(ACTIVE))=M3(I,1,J)
  570     CONTINUE
          GOTO 587
  572             CONTINUE
  573 CONTINUE
      GO TO 374
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  RETURN TO MAIN
      ELSE IF (EXECUTE .EQ. 9) THEN
      GOTO 143
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  REDRAW
C
C    XD         STORAGE OF TWO DIMENSIONAL DRAWING CORNER POINTS
C
C    COUNT      COUNTER FOR LINK DRAWING FRAMES
C
      ELSE IF (EXECUTE .EQ. 0) THEN
  587     CALL TRANSFILL(GO,TR,1,LINKNUM(1),1)
          IF (DRAW(1) .EQV. .TRUE.) THEN
              CALL XDFILL(XO,XD,TR,LINKNUM(1),1,
     $             FOCUS,MAG,COUNT,ZPOINT)
          END IF
C  THE TRANSFORMATION MATRIX OF THE LAST LINK ON THE MERLIN MUST BE
C  ATTACHED TO THE FIRST TRANSFORMATION MATRIX ON THE HAND, AND THE
C  FINGERS MUST BE ATTACHED TO THE HAND
      DO 592 L = 2,3
          IF (DRAW(L) .EQV. .TRUE.) THEN
            CALL ADD_ON(GO,TR,LINKNUM,L,RECEIVE)
            CALL TRANSFILL(GO,TR,2,LINKNUM(L),L)
            CALL XDFILL(XO,XD,TR,LINKNUM(L),L,
     $           FOCUS,MAG,COUNT,ZPOINT)
          END IF
  592 CONTINUE
      DO 598 N = 4,5
        IF (DRAW(N) .EQV. .TRUE.) THEN
          DO 594 L=HANDNUM(N-3),(HANDNUM(N-3)+4)
            CALL ADD_ON(GO,TR,LINKNUM,L,RECEIVE)
            CALL TRANSFILL(GO,TR,2,LINKNUM(L),L)
            CALL XDFILL(XO,XD,TR,LINKNUM(L),L,
     $         FOCUS,MAG,COUNT,ZPOINT)
  594     CONTINUE
```

```
        END IF
  598 CONTINUE
  604 CONTINUE
C
C  IF THERE IS ONLY ONE VIEW - DRAW VIEW
C
      IF (REPEAT .EQV. .FALSE.) THEN
       CALL DRAWBOT(XD,MONITOR,COUNT)
       COUNT = 0
       ELSE
C
C  IF MORE THAN ONE VIEW, DOES THE USER WANT TO ERASE
C  BETWEEN VIEWS? IF TO BE ERASED SET FLAG.(-999)
C  ADVANCE COUNT.
C
       IF (ERASE .EQV. .TRUE.) THEN
          COUNT = COUNT + 1
          XD(1,1,COUNT) = -999
       END IF
      END IF
C
C  RETURN TO SECTION OF PROGRAM FROM WHICH REDRAW WAS CALLED
C
      IF (OPTION .EQ. 1) THEN
          GO TO 157
          ELSE IF (OPTION .EQ. 2) THEN
               IF (EXECUTE .EQ. 6) THEN
                    GO TO 572
               ELSE
                    GO TO 374
               END IF
          ELSE
          GO TO 143
      END IF
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  QUIT
      ELSE IF (EXECUTE .EQ. 10) THEN
      GOTO 656
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  RETURN TO EXECUTE MENU
C
      ELSE
      GO TO 374
      END IF
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      ELSE IF (OPTION .EQ. 0) THEN
      GOTO 587
C  QUIT
      ELSE IF (OPTION .EQ. 10) THEN
      GOTO 656
      ELSE
      GO TO 143
      END IF
```

```
C  END OPTION
  656 CONTINUE
      STOP
      END
C  END OF MAIN PROGRAM
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     TR          STORES TRANSFORM MATRICES FOR ALL LINKS
C
C     ADD_ON      LINKS THE TRANSFORMATION MATRICES OF SYSTEMS
C
      SUBROUTINE ADD_ON(GO,TR,LINKNUM,ROBNUM,RECEIVE)
      REAL GO(3,2,10,10),TR(4,4,10,10)
      INTEGER ROBNUM,LINKNUM(20),RECEIVE(20)
      DO 15 J=1,4
        DO 15 I=1,4
          TR(I,J,1,ROBNUM)=TR(I,J,LINKNUM(RECEIVE(ROBNUM)),RECEIVE
     $          (ROBNUM))
   15 CONTINUE
      CALL TRANSFILL(GO,TR,2,LINKNUM(ROBNUM),ROBNUM)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C TRANSFILL FILLS TRANSFORMATION MATRIX
C
      SUBROUTINE TRANSFILL(GO,TR,BEG,END,R)
      REAL GO(3,2,10,14),TR(4,4,10,14)
      INTEGER BEG,END,R,ZPOINT,MONITOR
      DO 29 I=BEG,END
          TR(1,1,I,R)=COS(GO(3,2,I,R))*COS(GO(2,2,I,R))
          TR(1,2,I,R)=-SIN(GO(3,2,I,R))*COS(GO(2,2,I,R))
          TR(1,3,I,R)=SIN(GO(2,2,I,R))
          TR(1,4,I,R)=GO(1,1,I,R)
          TR(2,1,I,R)=COS(GO(3,2,I,R))*SIN(GO(2,2,I,R))*
     $         SIN(GO(1,2,I,R))+SIN(GO(3,2,I,R))*COS(GO(1,2,I,R))
          TR(2,2,I,R)=-SIN(GO(3,2,I,R))*SIN(GO(2,2,I,R))*
     $         SIN(GO(1,2,I,R))+COS(GO(3,2,I,R))*COS(GO(1,2,I,R))
          TR(2,3,I,R)=-COS(GO(2,2,I,R))*SIN(GO(1,2,I,R))
          TR(2,4,I,R)=GO(2,1,I,R)
          TR(3,1,I,R)=-COS(GO(3,2,I,R))*SIN(GO(2,2,I,R))*
     $         COS(GO(1,2,I,R))+SIN(GO(3,2,I,R))*SIN(GO(1,2,I,R))
          TR(3,2,I,R)=SIN(GO(3,2,I,R))*SIN(GO(2,2,I,R))*
     $         COS(GO(1,2,I,R))+COS(GO(3,2,I,R))*SIN(GO(1,2,I,R))
          TR(3,3,I,R)=COS(GO(1,2,I,R))*COS(GO(2,2,I,R))
          TR(3,4,I,R)=GO(3,1,I,R)
          TR(4,1,I,R)=0
          TR(4,2,I,R)=0
          TR(4,3,I,R)=0
          TR(4,4,I,R)=1
   29 CONTINUE
      CALL TRANSFORM(TR,1,END,R)
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  TRANSFORM CALCULATES THE TRANSFORMATION MATRIX BY MATRIX
C  MULTIPLICATION
C
      SUBROUTINE TRANSFORM(TR,SYS,END,R)
      DIMENSION TR(4,4,10,14),MATA(4,4),MATB(4,4),MATC(4,4)
      REAL TR,MATA,MATB,MATC
      INTEGER SYS,END,R
      DO 23 I = (SYS+1),END
        DO 15 K = 1,4
          DO 15 J = 1,4
               MATA(J,K) = TR(J,K,(I-1),R)
               MATB(J,K) = TR(J,K,I,R)
  15    CONTINUE
        CALL MULMATMAT(MATA,MATB,MATC)
        DO 22 K=1,4
          DO 22 J=1,4
             TR(J,K,I,R) = MATC(J,K)
  22      CONTINUE
  23  CONTINUE
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  XDFILL FILLS THE XD ARRAY (8 X 2 X COUNT) WITH DRAWING POINTS
C  POSITIONS
C
      SUBROUTINE XDFILL(XO,XD,TR,END,R,FOCUS,MAG,COUNT,ZPOINT)
      DIMENSION XO(8,3,10,14),TR(4,4,10,14),MAT(4,4),
     $  VEC(4),CORD(4),XD(8,2,500),FOCUS(2)
      REAL XO,TR,MAT,VEC,CORD,X,Y,Z,XD,THX,THY,ZPOINT,FOCUS,MAG
      INTEGER END,R,COUNT
      DO 43 L=1,END
          COUNT = COUNT + 1
          DO 42 I=1,8
             DO 18 K=1,4
               DO 18 J=1,4
                  MAT(J,K) = TR(J,K,L,R)
  18         CONTINUE
             DO 21 J=1,3
                VEC(J)=XO(I,J,L,R)
  21         CONTINUE
             VEC(4)=1.0
C
C  MULMATVEC IS A MATRIX OPERATION OF MULTIPLYING THE VECTOR BY THE
C  TRANSFORMATION MATRIX
C
             CALL MULMATVEC(MAT,VEC,CORD)
C
C THE SCREEN VIEW ALLOWS A 20 DEGREE VIEWING TUNNEL
C THE FOLLOWING CODE PLACES THE DRAWING POINTS IN THEIR RESPECTIVE
C PLACES IN THE SCREEN VIEW. IF THE POINTS LIE OUTSIDE THE
```

```
C 20 DEGREE TUNNEL, THEY WILL NOT BE DISPLAYED
C
              X = CORD(1) - FOCUS(1)
              Y = CORD(2) - FOCUS(2)
              Z = ZPOINT - CORD(3)
              THX=ATAN(X/Z)
              THY=ATAN(Y/Z)
              XD(I,1,COUNT) = 50*THX/.349*MAG
              XD(I,2,COUNT) = 50*THY/.349*MAG
  42        CONTINUE
  43    CONTINUE
        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C MULMATVEC MULTIPLIES A 4X4 MATRIX WITH A 4X1 VECTOR OUTPUT C(4X4)
C
        SUBROUTINE MULMATVEC(A,B,C)
        REAL A(4,4),B(4),C(4),SUM
        DO 14 I = 1,4
           SUM = 0
             DO 12 J = 1,4
                SUM = A(I,J)*B(J) + SUM
  12         CONTINUE
             C(I) = SUM
  14    CONTINUE
        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C MULMATMAT MULTIPLIES TWO 4X4 MATRICES, WITH OUTPUT C(4X4)
C
        SUBROUTINE MULMATMAT(A,B,C)
        REAL A(4,4),B(4,4),C(4,4),SUM
        DO 16 I = 1,4
          DO 15 J = 1,4
              SUM = 0
              DO 13 K = 1,4
                SUM = A(I,K)*B(K,J) + SUM
  13          CONTINUE
              C(I,J) = SUM
  15      CONTINUE
  16    CONTINUE
        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  SUBROUTINE GENANG GENERATES AN ARRAY OF ANGLES.
C  GIVEN AN INITIAL SET OF ANGLES, A FINAL SET OF ANGLES, AND THE
C  NUMBER OF STEPS, IT COMPUTES EQUALLY SPACED INTERMEDIATE ANGLES,
C  GOING FROM THE INITIAL TO THE FINAL ANGLE.
C
        SUBROUTINE GENANG(M1,M2,N1,N2,M3,STEPS)
        REAL M1(6,4),M2(6,4),M3(6,4,200),DELM(6,4)
```

```
      INTEGER N1,N2,STEPS
      WRITE(6,*)'INPUT - NUMBER OF STEPS'
      READ(6,*) STEPS
      DO 11 J=1,N2
          DO 11 I=1,N1
          DELM(I,J)=(M2(I,J)-M1(I,J))/STEPS
  11  CONTINUE
      DO 15 K=1,STEPS
          DO 15 J=1,N2
               DO 15 I=1,N1
               M3(I,J,K)=M1(I,J)+DELM(I,J)*K
  15  CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  SUBROUTINE DRAWBOT CONTAINS THE DISSPLA SETUP CALLS.
C  IT RECEIVES THE TWO DIMENSIONAL SCREEN DRAWING POINTS.
C  EACH SET OF EIGHT POINTS OF A LINK ARE CONNECTED IN A CERTAIN
C  SEQUENCE TO REPRESENT A SIX SIDED CLOSED OBJECT (THE LINK).
C
      SUBROUTINE DRAWBOT(XD,MONITOR,COUNT)
      REAL XD(8,2,1000),X(16),Y(16)
      INTEGER MONITOR,B(16),COUNT
      DATA B/1,2,4,3,7,5,6,2,6,8,4,8,7,5,1,3/
C
C SYSTEM MODULE CALL
C
      IF (MONITOR .EQ. 1) THEN
          CALL TEKALL(4010,960,0,1,0)
      ELSE IF (MONITOR .EQ. 2) THEN
          CALL TK41(4107)
      ELSE
          CALL REGIS(3,0)
      ENDIF
C
C THE FOLLOWING ARE DISSPLA DRAWING COMMANDS
C
      CALL BLOWUP(1.0)
      CALL NOCHEK
      CALL PAGE(11.0,11.0)
      CALL AREA2D(10.5,10.5)
      CALL GRAF(-50.0,'SCALE',50.0,-50.0,'SCALE',50.0)
      CALL INTAXS
      CALL SCLPIC(.01)
      DO 51 I=1,COUNT
C
C  IF FLAG (XD(1,1,COUNT)=-999) ERASE SCREEN AND DRAW NEXT VIEW
C
        IF (XD(1,1,I) .EQ. -999) THEN
          CALL ENDPL(0)
          CALL PAGE(11.0,11.0)
          CALL AREA2D(10.5,10.5)
          CALL GRAF(-50.0,'SCALE',50.0,-50.0,'SCALE',50.0)
```

```
          CALL INTAXS
          CALL SCLPIC(.01)
        ELSE
        DO 45 J=1,16
C
C  FILL THE DRAWING ARRAY IN A SEQUENCE THAT RESULTS IN DRAWING
C  A CUBE WHEN THE CONSECUTIVE POINTS ARE CONNECTED WITH LINES
C
          X(J)=XD(B(J),1,I)
          Y(J)=XD(B(J),2,I)
 45     CONTINUE
C
C  DRAW THE LINK.
C
        CALL CURVE(X,Y,16,1)
        ENDIF
 51   CONTINUE
      CALL ENDPL(0)
      CALL DONEPL
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

PROGRAM INKIN.FOR

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  MERLIN ROBOT INVERSE KINEMATICS PROGRAM
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE INKIN(ANG,HAND)
C  DEFINE REAL & INTEGER VALUES
      INTEGER FLAG,WSPACE,HAND,SET
      REAL T(4,4),T1,T2P1,T2P2,T3P,T3N,T4A,T4B,T5A,T5B,T6A,
     $T6B,A2,D2,D3,D4,D6,PI,WP,S1,S2,EP,EN,WR1,WR2,WR3,WR4,
     $WP1,WP2,WP3,WP4,HR1,HR2,HR3,HR4,DUM,Z(4,7),TPP,TPN,ANG(1,6)
 1    PRINT *,'  '
C  VALUE OF CONSTANTS
      PI = 3.141592653589792
C  SETUP KINEMATIC PARAMETERS FOR THE MERLIN 6500 50 LB. ROBOT
C
C  A2 IS THE DISTANCE BETWEEN SHOULDER JOINT AND ELBOW JOINT
      A2 = 17.38
C  D4 IS THE DISTANCE FROM ELBOW JOINT TO WRIST PIN
      D4 = 17.24
C  D6 IS THE DISTANCE FROM WRIST PIN TO TIP OF THE END-EFFECTOR
      D6 = 3.5
C  SET UP D2 AND D3. D2 IS THE DISTANCE FROM THE WAIST VERTICAL
C  AXIS TO THE CENTER OF THE UPPER ARM. D3 IS THE DISTANCE FROM
C  THE CENTER OF THE UPPER ARM TO THE CENTER OF THE LOWER ARM.
C  FOR LEFT HAND.
      IF(HAND .EQ. 1) THEN
```

```
        D2 = 19.00
        D3 = -7.00
      ELSE
C  SET UP D2 AND D3 FOR THE RIGHT HAND.
        D2 = -19.00
        D3 = 7.00
      ENDIF
C  INITIALIZE ALL GLOBAL VARIABLES ( RETURNED VARIABLES ARE
C  INITIALIZED INSIDE THE SUBROUTINE ONLY )
      WP = 0.0
      S1 = 0.0
      S2 = 0.0
      EP = 0.0
      EN = 0.0
      WR1 = 0.0
      WR2 = 0.0
      WR3 = 0.0
      WR4 = 0.0
      WP1 = 0.0
      WP2 = 0.0
      WP3 = 0.0
      WP4 = 0.0
      HR1 = 0.0
      HR2 = 0.0
      HR3 = 0.0
      HR4 = 0.0
      DUM = 0.0
      T2P1 = 0.0
      T2P2 = 0.0
C  INITIALIZE [ Z ] MATRIX
C  THE FIRST COLUMN OF THE MATRIX IS A FLAG FOR VALIDITY OF THE
C  SET OF JOINT ANGLES BEING COMPUTED BEING ALL WITHIN THEIR RANGES.
C  THE REMAINING 4 X 6 MATRIX IS USED TO STORE THE RESULTS OF
C  THE COMPUTATIONS IN THE ORDER -- WAIST, SHOULDER, ELBOW,
C  WRIST ROLL, WRIST PITCH, HAND ROLL
      DO 2 I = 1,4
        DO 2 J = 1,7
          Z(I,J) = 0.0
 2    CONTINUE
C  ENTER POSITION AND ORIENTATION MATRIX FROM DATAFILE OR SCREEN
 3    CALL MATENTER(T,D6)
C  FLAG SET UP FOR END POSITION IN/OUT OF WORKSPACE
C  WSPACE = 0 IF THE END-EFFECTOR IS INSIDE THE WORKSPACE
C  WSPACE = 1 IF THE END-EFFECTOR IS OUTSIDE THE WORKSPACE
C  SET DEFAULT WSPACE FLAG = 0
      WSPACE = 0
C  COMPUTE WAIST ANGLES T1.
C  IN THE CALL STATEMENT BELOW, T IS THE 4 X 4 POSITION AND
C  ORIENTATION WORKSPACE, T1 IS THE COMPUTED WAIST ANGLE.
      CALL WAIST(T,T1,D2,D3,WSPACE)
C  IF POSITION DESIRED AS END-POINT IS OUTSIDE THE WORKSPACE
C  GET A NEW SET OF ENDPOINTS FROM THE USER.
      IF(WSPACE .EQ. 1) THEN
        GOTO 3
```

```
      ENDIF
C  CONVERT COMPUTED WAIST ANGLE FROM RADIANS TO DEGREES.
C  A DUMMY IS USED HERE (DUM), SINCE ONLY ONE WAIST ANGLE EXISTS.
      CALL RADEG(T1,0.0,WP,DUM)
C  STORE RESULTS OF WAIST IN [Z] MATRIX (SECOND COLUMN)
      DO 5 I = 1,4
        Z(I,2) =  WP
 5    CONTINUE
C  RESET THE WSPACE FLAG TO 0 FOR THE ELBOW COMPUTATIONS.
      WSPACE = 0
C  COMPUTE ELBOW ANGLES T3P,T3N
      CALL ELBOW(T,T3P,T3N,A2,D2,D3,D4,WSPACE)
C  IF USER DEFINED END POSITION AND ORIENTATION IS OUTSIDE
C  THE WORKSPACE, RE-ENTRY OF MATRIX
      IF(WSPACE .EQ. 1) THEN
        GOTO 3
      ENDIF
C  CONVERT ELBOW ANGLES FROM RADIANS TO DEGREES
      CALL RADEG(T3P,T3N,EP,EN)
C  STORE RESULTS OF ELBOW ANGLE SOLUTION IN THE FOURTH COLUMN
C  OF MATRIX [Z]
      DO 6 I = 1,2
        Z(I,4) = EP
        Z(I+2,4) = EN
 6    CONTINUE
C  COMPUTE ( SHOULDER + ELBOW ) ANGLES TPP,TPN
      CALL SHOULDER(T,A2,D4,T1,T3P,T3N,TPP,TPN)
C  COMPUTE SHOULDER ANGLES T2P1,T2P2
      T2P1 = TPP - T3P
      T2P2 = TPN - T3N
C  CONVERT SHOULDER ANGLES FROM RADIANS TO DEGREES
      CALL RADEG(T2P1,T2P2,S1,S2)
C  STORE RESULTS OF SHOULDER ANGLES IN [Z] MATRIX (THIRD COLUMN)
      DO 7 I = 1,2
        Z(I,3) = S1
        Z(I+2,3) = S2
 7    CONTINUE
C  COMPUTE WRIST ROLL ANGLES
      CALL WROLL(T,T4P1,T4P2,T1,TPP,TPN)
C  CONVERT WRIST ROLL ANGLES FROM RADIANS TO DEGREES
      CALL RADEG(T4P1,T4P2,WR1,WR2)
C  COMPUTE 'WRIST FLIPPED' SOLUTIONS
        WR3 = WR1 + 180.0
        WR4 = WR2 + 180.0
C  STORE RESULTS OF WRIST ROLL IN [Z] MATRIX (FIFTH COLUMN)
      Z(1,5) = WR1
      Z(2,5) = WR3
      Z(3,5) = WR2
      Z(4,5) = WR4
C  COMPUTE WRIST PITCH ANGLES
      CALL WPITCH(T,T5P1,T5P2,T1,TPP,TPN,T4P1,T4P2)
C  CONVERT WRIST PITCH ANGLES FROM RADIANS TO DEGREES
      CALL RADEG(T5P1,T5P2,WP1,WP2)
```

```
C  COMPUTE 'REVERSED PITCH' SOLUTIONS
      VP3 = - VP1
      VP4 = - VP2
C  STORE RESULTS IN  [ Z ] MATRIX - SIXTH COLUMN
      Z(1,6) = VP1
      Z(2,6) = VP3
      Z(3,6) = VP2
      Z(4,6) = VP4
C  COMPUTE HAND ROLL
      CALL HROLL(T,T6P1,T6P2,T1,TPP,TPN,T4P1,T4P2,T5P1,T5P2)
C  CONVERT HAND ROLL ANGLES FROM RADIANS TO DEGREES
      CALL RADEG(T6P1,T6P2,HR1,HR2)
C  COMPUTE 'HAND FLIPPED' SOLUTIONS FOR HAND ROLL
      HR3 = HR1 + 180.0
      HR4 = HR2 + 180.0
C  STORE RESULTS IN THE [Z] MATRIX (SEVENTH COLUMN)
      Z(1,7) = HR1
      Z(2,7) = HR3
      Z(3,7) = HR2
      Z(4,7) = HR4
C  NORMALIZE THE COMPUTED RESULTS.
      CALL NORMAL(Z)
C  CHECK FOR VALIDITY OF EACH SOLUTION SET.
      CALL VALID(Z)
C  PRINT OUT VALID RESULTS ( VALID IF WITHIN JOINT ANGLE RANGE )
      PRINT *,' THE VALID INVERSE KINEMATICS RESULTS ARE :== '
      DO 51 I = 1,4
        IF(Z(I,1) .EQ. 0.0) THEN
          WRITE(5,*) 'THE VALID SOLUTION NUMBER IS ',I
          WRITE(5,*) (Z(I,J),J=2,7)
        ENDIF
 51   CONTINUE
C  SET DEFAULT FOR USER CHOSEN SET OF RESULTS = 1
      SET = 1
C  QUERY USER FOR CHOICE OF SET OF RESULTS FROM VALID SET.
      PRINT *,' ENTER YOUR CHOICE OF ANGLES (AS A SET) '
      PRINT *,' FROM THE DIFFERENT SETS ABOVE '
      READ(5,52) SET
 52   FORMAT(I)
C  COPY CHOSEN SET OF RESULTS TO VECTOR ANG(1,6)
      DO 53 J = 1,6
        ANG(1,J) = Z(SET,J+1)
 53   CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  POSITION AND ORIENTATION DATA ENTRY ROUTINE
      SUBROUTINE MATENTER(A,D6)
      INTEGER MATCH,TIP
      REAL A(4,4),PX1,PY1,PZ1,PX,PY,PZ,D6
C  INITIALIZE LOCAL VARIABLES
      PX1 = 0.0
      PY1 = 0.0
      PZ1 = 0.0
```

```
      PX  = 0.0
      PY  = 0.0
      PZ  = 0.0
C  INITIALIZE MATRIX [A]
 100  DO 101 I = 1,4
        DO 101 J = 1,4
          A(I,J) = 0.0
 101  CONTINUE
C   DATA ENTRY OF POSITION AND ORIENTATION MATRIX
        DO 102 I = 1,3
          DO 102 J = 1,4
            PRINT *,' ENTER TRANSFORM MATRIX ENTRY',I,J
            READ(5,*) A(I,J)
 102    CONTINUE
C  ADJUST ROW 4 ENTRIES TO PREVENT ENTRY ERROR
        A(4,1) = 0.0
        A(4,2) = 0.0
        A(4,3) = 0.0
        A(4,4) = 1.0
C  PRINT OUT MATRIX TO SCREEN
      PRINT *,' '
      PRINT *,' THIS IS THE ENTERED TRANSFORM MATRIX. '
      CALL AOUT(A)
      PRINT *,' IF YOU WANT TO CHANGE THE MATRIX, ENTER 0 '
      PRINT *,' IF POSITION ENTRIES REFER TO THE TIP OF THE '
      PRINT *,' END  EFFECTOR  -----  ENTER  1 '
      PRINT *,' IF POSITION ENTRIES ARE WITH RESPECT TO THE '
      PRINT *,' WRIST  PIN  ----------  ENTER 2 '
      READ(5,104) TIP
 104  FORMAT(I)
C  ALLOW FOR CHANGE OF TRANSFORM MATRIX ENTRIES
      IF(TIP .EQ. 0) THEN
        GOTO 100
C  ADJUST END EFFECTOR POSITION TO WRIST PIN IF POSITION GIVEN IS
C  AT THE TIP OF THE END-EFFECTOR
      ELSEIF(TIP .EQ. 1) THEN
C  SETUP POSITION PARAMETERS TO END-EFFECTOR TIP
        PX1 = A(1,4)
        PY1 = A(2,4)
        PZ1 = A(3,4)
C  ADJUST POSITION PARAMETERS TO WRIST PIN
        PX = PX1 - D6 * A(1,3)
        PY = PY1 - D6 * A(2,3)
        PZ = PZ1 - D6 * A(3,3)
C  RESET POSITION PARAMETERS IN [A] MATRIX TO WRIST PIN
        A(1,4) = PX
        A(2,4) = PY
        A(3,4) = PZ
      ENDIF
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
C  OUTPUT OF MATRIX TO SCREEN
      SUBROUTINE AOUT(M)
      REAL M(4,4)
      INTEGER I,J
      DO 1001 I = 1,4
        WRITE(5,*) (M(I,J),J=1,4)
 1001   CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  WAIST ANGLE COMPUTATION
      SUBROUTINE WAIST(A,W1,X2,X3,SPACE)
      REAL A(4,4),W1,X2,X3,RHO,PX,PY,TERM1,TERM2,T2,X23
      INTEGER I,J,SPACE
C  INITIALIZATION OF LOCAL VARIABLES
      W1 = 0.0
      TERM1 = 0.0
      TERM2 = 0.0
      X23 = 0.0
      T2 = 0.0
C  SET UP OF POSITION PARAMETERS
      PX = A(1,4)
      PY = A(2,4)
C  COMPUTE FIRST TERM FOR WAIST ANGLE SOLUTION
      TERM1 = ATAN2(PY,PX)
C  COMPUTE TERM2
      X23 = (X2 + X3)
      PXSQ = PX * PX
      PYSQ = PY * PY
      PXPYSQ = PXSQ + PYSQ
      X23SQ = X23 * X23
C  IS USER-SPECIFIED POSITION INSIDE THE WORKSPACE ?
C  SET WORKSPACE FLAG TO INSIDE WORKSPACE
      SPACE = 0
      IF(PXPYSQ .GT. X23SQ) THEN
C  SPECIFIED POSITION IS INSIDE WORK-SPACE, SO COMPUTE SECOND TERM
        GOTO 301
      ELSE
C  USER SPECIFIED POSITION IS OUTSIDE WORKSPACE.
C
C  COMPUTE DIFFERENCE IN TERMS
        ERROR = (ABS(PXPYSQ - X23SQ))
C  IF THE COMPUTED ERROR < 0.0001, THEN COMPUTATIONAL ERROR
C  COULD HAVE CAUSED THE POSITION TO LIE OUTSIDE THE WORKSPACE.
        IF(ERROR .LT. 0.0001) THEN
C  YES, COMPUTATIONAL ERROR OCCURED. COMPUTE T2, FOLLOWED BY
C  THE SECOND TERM.
          T2 = SQRT(ERROR)
          GOTO 302
        ELSE
C  USER SPECIFIED POSITION IS DEIFINITELY OUT OF WORKSPACE
          SPACE = 1
          PRINT *,' OUTSIDE WORKSPACE '
          GOTO 303
```

```
      ENDIF
      ENDIF
 301  T2 = SQRT(PXPYSQ - X23SQ)
 302  TERM2 = ATAN2(X23,T2)
C  COMPUTE SOLUTION FOR WAIST ANGLE W1
      W1 = TERM1 - TERM2
 303  RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  ELBOW ANGLE DETERMINATION ROUTINE
      SUBROUTINE ELBOW(A,EP,EN,B2,X2,X3,X4,SPACE)
      INTEGER SPACE
      REAL A(4,4),EP,EN,B2,X2,X3,X4,KA,KB,X23,T1,T2P,T2N
C  INITIALIZE LOCAL VARIABLES
      EP = 0.0
      EN = 0.0
      KA = 0.0
      KB = 0.0
      T1 = 0.0
      T2P = 0.0
      T2N = 0.0
      X23 = 0.0
C  SET UP POSITION PARAMETERS OF TRANSFORM MATRIX
      PX = A(1,4)
      PY = A(2,4)
      PZ = A(3,4)
C  COMPUTE FIRST TERM OF ARCTAN FUNCTION
      X23 = ( X2 + X3 )
      KA = - (PX * PX) - (PY * PY) - (PZ * PZ)
      KB = (B2 * B2) + (X23 * X23) + (X4 * X4)
      T1 = (KA + KB) / ( 2.0 * B2 * X4)
      T1SQ = T1 * T1
C  DETERMINE IF USER DEFINED POSITION IS OUTSIDE WORKSPACE
      SPACE = 0
C  POSITION IS INSIDE THE WORK-SPACE IF T1SQ < 1.0
      IF(T1SQ .LE. 1.0) THEN
        GOTO 401
      ELSE
C  USER DEFINED POSITION MAYBE OUTSIDE WORKSPACE
C  THEREFORE, COMPUTE THE ERROR
        ERROR = (ABS(1.0 - T1SQ))
C  CHECK TO SEE IF COMPUTATIONAL ERROR COULD HAVE CAUSED THE
C  POSITION TO LIE OUTSIDE THE WORKSPACE
        IF(ERROR .LT. 0.0001) THEN
          T2P = SQRT(ERROR)
          GOTO 402
        ELSE
C  USER ENTERED POSITION IS OUTSIDE WORKSPACE
          PRINT *,' OUTSIDE WORKSPACE '
          SPACE = 1
          GOTO 403
        ENDIF
      ENDIF
C  COMPUTE SECOND TERM OF ARCTAN FUNCTION
```

```
 401  T2P = SQRT(1.0 - T1SQ)
 402  T2N = - T2P
C  COMPUTE THE TWO POSSIBLE SOLUTIONS FOR ELBOW ANGLE I.E. EP & EN
      EP = ATAN2(T1,T2P)
      EN = ATAN2(T1,T2N)
 403  RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  SHOULDER + ELBOW ANGLE DETERMINATION ROUTINE
      SUBROUTINE SHOULDER(A,B2,X4,WP,EP,EN,APP,APN)
      INTEGER I,J
      REAL A(4,4),B2,X4,WP,EP,EN,T1PP,T1PN,T2PP,T2PN,C1P,S1P
     $C3P,C3N,S3P,S3N,T1PPA,T1PNA,T2PPB,T2PNB,APP,APN
C  INITIALIZE LOCAL VARIABLES
      T1PP = 0.0
      T1PN = 0.0
      T2PP = 0.0
      T2PN = 0.0
      T1PPA = 0.0
      T1PNA = 0.0
      T1PP = 0.0
      T1PN = 0.0
      T2PPB = 0.0
      T2PNB = 0.0
      APP = 0.0
      APN = 0.0
C  SETUP OF POSITION PARAMETERS
      PX = A(1,4)
      PY = A(2,4)
      PZ = A(3,4)
C  COMPUTE COSINE AND SINE FUNCTION VALUES OF THE APPROPRIATE ANGLES
      C1P = COS(WP)
      S1P = SIN(WP)
      C3P = COS(EP)
      S3P = SIN(EP)
      C3N = COS(EN)
      S3N = SIN(EN)
C  COMPUTE ALL POSSIBLE FIRST TERMS OF ARCTAN2 FUNCTION
C
C  WAIST POSITIVE, ELBOW POSITIVE (T1PP)
      T1PPA = B2 * C3P * PZ
      T1PP = (((B2 * S3P) - X4) * ((C1P * PX) + (S1P * PY))) - T1PPA
C  WAIST POSITIVE, ELBOW NEGATIVE (T1PN)
      T1PNA = B2 * C3N * PZ
      T1PN = (((B2 * S3N) - X4) * ((C1P * PX) + (S1P * PY))) - T1PNA
C  COMPUTE ALL POSSIBLE SECOND TERMS OF ARCTAN2 FUNCTION
C
C  WAIST POSITIVE, ELBOW POSITIVE (T2PP)
      T2PPB = ((B2 * C3P) * ((C1P * PX) + (S1P * PY)))
      T2PP = (((B2 * S3P) - X4) * PZ) + T2PPB
C  WAIST POSITIVE, ELBOW NEGATIVE (T2PN)
      T2PNB = ((B2 * C3N) * ((C1P * PX) + (S1P * PY)))
      T2PN = (((B2 * S3N) - X4) * PZ) + T2PNB
C  COMPUTE ALL FOUR POSSIBLE SOLUTIONS OF (THETA 2 + THETA 3)
```

```
      APP = ATAN2(T1PP,T2PP)
      APN = ATAN2(T1PN,T2PN)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  WRIST ROLL ANGLE DETERMINATION ROUTINE
      SUBROUTINE WROLL(A,PPP,PPN,WP,T23PP,T23PN)
      INTEGER FPPP,FPPN
      REAL A(4,4),PPP,PPN,WP,T23PP,T23PN,T1P,R13,R23,R33,
     $S1P,C1P,C23PP,C23PN,S23PP,S23PN,SNGCHK
C  INITIALIZE LOCAL VARIABLES
      T1P = 0.0
      T2PPP = 0.0
      T2PPN = 0.0
      PPP = 0.0
      PPN = 0.0
C  SET UP SINGULARITY CHECK CONDITION
      SNGCHK = 0.005
C  SET FLAGS TO NON-SINGULAR CASE
      FPPP = 0
      FPPN = 0
C  SETUP MATRIX ORIENTATION PARAMETERS
      R13 = A(1,3)
      R23 = A(2,3)
      R33 = A(3,3)
C  SETUP TRIG. FUNCTIONS
      S1P = SIN(WP)
      C1P = COS(WP)
      C23PP = COS(T23PP)
      S23PP = SIN(T23PP)
      C23PN = COS(T23PN)
      S23PN = SIN(T23PN)
C  COMPUTE ALL FIRST TERMS OF ARCTAN2 FUNCTION
      T1P = - (R13 * S1P) + (R23 * C1P)
C  COMPUTE ALL SECOND TERMS OF ARCTAN2 FUNCTION
      T2PPP = -(R13*C1P*C23PP) - (R23*S1P*C23PP) + (R33*S23PP)
      T2PPN = -(R13*C1P*C23PN) - (R23*S1P*C23PN) + (R33*S23PN)
C  CHECK FOR SINGULARITY CONDITIONS AT WRIST PITCH
      IF((T1P .LT. SNGCHK .AND. T1P .GT. - SNGCHK) .AND.
     $(T2PPP .LT. SNGCHK .AND. T2PPP .GT.- SNGCHK)) THEN
      FPPP = 1
      ENDIF
      IF((T1P .LT. SNGCHK .AND. T1P .GT. - SNGCHK) .AND.
     $(T2PPN .LT. SNGCHK .AND. T2PPN .GT. - SNGCHK)) THEN
      FPPN = 1
      ENDIF
C  SET WRIST ROLL TO 0.0 RADIANS IF SINGULARITY DETECTED
C  AT WRIST PITCH, ELSE COMPUTE WRIST ROLL. NOTE THAT THIS WILL
C  CAUSE THE ROLL TO SHOW UP ONLY IN HAND ROLL ANGLE.
C  SOLUTION # 1
      IF(FPPP .EQ. 1) THEN
        PPP = 0.0
      ELSE
        PPP = ATAN2(T1P,T2PPP)
```

```
      ENDIF
C  SOLUTION # 2
      IF(FPPN .EQ. 1) THEN
        PPN = 0.0
      ELSE
        PPN = ATAN2(T1P,T2PPN)
      ENDIF
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  WRIST PITCH DETERMINATION
      SUBROUTINE WPITCH(A,A5P1,A5P2,WP,APP,APN,W4P1,W4P2)
      REAL A(4,4),A5P1,A5P2,WP,APP,APN,W4P1,W4P2,
     $T5A1PPPP,T5A2PPPP,T5A3PPPP,T5APPPP,T5A1PPNP,T5A2PPNP,T5A3PPNP,
     $T5APPNP,T5B1PPPP,T5B2PPPP,T5B3PPPP,T5BPPPP,T5B1PPNP,T5B2PPNP,
     $T5B3PPNP,T5BPPNP,R13,R23,R33,C1P,S1P,C23PP,S23PP,
     $C23PN,S23PN,C4P1,S4P1,C4P2,S4P2
C  INITIALIZE LOCAL VARIABLES TO 0.0
      T5A1PPPP = 0.0
      T5A2PPPP = 0.0
      T5A3PPPP = 0.0
      T5APPPP  = 0.0
      T5A1PPNP = 0.0
      T5A2PPNP = 0.0
      T5A3PPNP = 0.0
      T5APPNP  = 0.0
      T5B1PPPP = 0.0
      T5B2PPPP = 0.0
      T5B3PPPP = 0.0
      T5BPPPP  = 0.0
      T5B1PPNP = 0.0
      T5B2PPNP = 0.0
      T5B3PPNP = 0.0
      T5BPPNP  = 0.0
      A5P1 = 0.0
      A5P2 = 0.0
C  SETUP ORIENTATION PARAMETERS
      R13 = A(1,3)
      R23 = A(2,3)
      R33 = A(3,3)
C  SETUP TRIG. FUNCTIONS
      C1P = COS(WP)
      S1P = SIN(WP)
      C23PP = COS(APP)
      S23PP = SIN(APP)
      C23PN = COS(APN)
      S23PN = SIN(APN)
      C4P1 = COS(W4P1)
      S4P1 = SIN(W4P1)
      C4P2 = COS(W4P2)
      S4P2 = SIN(W4P2)
C  COMPUTE FIRST TERMS OF THE ARCTAN2 FUNCTIONS
      T5A1PPPP = - (R13 * ((C1P * C23PP * C4P1) + (S1P * S4P1)))
      T5A2PPPP = - (R23 * ((S1P * C23PP * C4P1) - (C1P * S4P1)))
```

```
      T5A3PPPP = R33 * S23PP * C4P1
      T5APPPP = T5A1PPPP + T5A2PPPP + T5A3PPPP
      T5A1PPNP = - (R13 * ((C1P * C23PN * C4P2) + (S1P * S4P2)))
      T5A2PPNP = - (R23 * ((S1P * C23PN * C4P2) - (C1P * S4P2)))
      T5A3PPNP = R33 * S23PN * C4P2
      T5APPNP = T5A1PPNP + T5A2PPNP + T5A3PPNP
C  COMPUTE SECOND TERMS OF THE ARCTAN2 FUNCTIONS
      T5B1PPPP = - (C1P * S23PP * R13)
      T5B2PPPP = - (S1P * S23PP * R23)
      T5B3PPPP = - (C23PP * R33)
      T5BPPPP = T5B1PPPP + T5B2PPPP + T5B3PPPP
      T5B1PPNP = - (C1P * S23PN * R13)
      T5B2PPNP = - (S1P * S23PN * R23)
      T5B3PPNP = - (C23PN * R33)
      T5BPPNP = T5B1PPNP + T5B2PPNP + T5B3PPNP
C  COMPUTE WRIST PITCH ANGLES USING ARCTAN2 FUNCTION
      A5P1 = ATAN2(T5APPPP,T5BPPPP)
      A5P2 = ATAN2(T5APPNP,T5BPPNP)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  DETERMINATION OF HAND ROLL ANGLES
      SUBROUTINE HROLL(A,A6P1,A6P2,WP,APP,APN,A4P1,A4P2,A5P1,A5P2)
      INTEGER I,J
      REAL A(4,4),A6P1,A6P2,WP,APP,APN,A4P1,A4P2,A5P1,A5P2,
     $PPPPPA1,PPPPPA2,PPPPPA3,PPPPPA,PPNPPA1,PPNPPA2,PPNPPA3,PPNPPA
     $PPPPPB1,PPPPPB2,PPPPPB3,PPPPPB,PPNPPB1,PPNPPB2,PPNPPB3,PPNPPB
C  INITIALIZE LOCAL VARIABLES TO 0.0
      PPPPPA1 = 0.0
      PPPPPA2 = 0.0
      PPPPPA3 = 0.0
      PPPPPA  = 0.0
      PPNPPA1 = 0.0
      PPNPPA2 = 0.0
      PPNPPA3 = 0.0
      PPNPPA  = 0.0
      PPPPPB1 = 0.0
      PPPPPB2 = 0.0
      PPPPPB3 = 0.0
      PPPPPB  = 0.0
      PPNPPB1 = 0.0
      PPNPPB2 = 0.0
      PPNPPB3 = 0.0
      PPNPPB  = 0.0
C  INITIALIZE WRIST ROLL ANGLES TO 0.0
      A6P1 = 0.0
      A6P2 = 0.0
C  SETUP ROTATION PARAMETERS
      R11 = A(1,1)
      R21 = A(2,1)
      R31 = A(3,1)
C  SETUP UP TRIG. FUNCTIONS
      C1P = COS(WP)
      S1P = SIN(WP)
```

```
      C23PP = COS(APP)
      S23PP = SIN(APP)
      C23PN = COS(APN)
      S23PN = SIN(APN)
      C4P1 = COS(A4P1)
      S4P1 = SIN(A4P1)
      C4P2 = COS(A4P2)
      S4P2 = SIN(A4P2)
      C5P1 = COS(A5P1)
      S5P1 = SIN(A5P1)
      C5P2 = COS(A5P2)
      S5P2 = SIN(A5P2)
C  COMPUTE THE FIRST TERMS FOR THE ARCTAN2 FUNCTION
      PPPPPA1 = R11 * ((C1P * C23PP * S4P1) - (S1P * C4P1))
      PPPPPA2 = R21 * ((S1P * C23PP * S4P1) + (C1P * C4P1))
      PPPPPA3 = R31 * (S23PP * S4P1)
      PPPPPA = - PPPPPA1 - PPPPPA2 + PPPPPA3
      PPNPPA1 = R11 * ((C1P * C23PN * S4P2) - (S1P * C4P2))
      PPNPPA2 = R21 * ((S1P * C23PN * S4P2) + (C1P * C4P2))
      PPNPPA3 = R31 * (S23PN * S4P2)
      PPNPPA = - PPNPPA1 - PPNPPA2 + PPNPPA3
C  COMPUTE THE SECOND TERMS FOR THE ARCTAN2 FUNCTION
      PPPPPB1 = R11 * (C5P1 * ((C1P * C23PP * C4P1) + (S1P * S4P1))
     $ - (C1P * S23PP * S5P1))
      PPPPPB2 = R21 * (C5P1 * ((S1P * C23PP * C4P1) - (C1P * S4P1))
     $ - (S1P * S23PP * S5P1))
      PPPPPB3 = R31 * ((S23PP * C4P1 * C5P1) + (C23PP * S5P1))
      PPPPPB = PPPPPB1 + PPPPPB2 - PPPPPB3
      PPNPPB1 = R11 * (C5P2 * ((C1P * C23PN * C4P2) + (S1P * S4P2))
     $ - (C1P * S23PN * S5P2))
      PPNPPB2 = R21 * (C5P2 * ((S1P * C23PN * C4P2) - (C1P * S4P2))
     $ - (S1P * S23PN * S5P2))
      PPNPPB3 = R31 * ((S23PN * C4P2 * C5P2) + (C23PN * S5P2))
      PPNPPB = PPNPPB1 + PPNPPB2 - PPNPPB3
C  COMPUTE THE HAND ROLL ANGLE USING THE ARCTAN2 FUNCTION
      A6P1 = ATAN2(PPPPPA,PPPPPB)
      A6P2 = ATAN2(PPNPPA,PPNPPB)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  RADIAN TO DEGREE CONVERSION ROUTINE
      SUBROUTINE RADEG(RAD1,RAD2,DEG1,DEG2)
      REAL RAD1,RAD2,DEG1,DEG2,PI
C  INITIALIZE LOCAL VARIABLES AND RETURNED VALUES
      DEG1 = 0.0
      DEG2 = 0.0
C  DECLARE CONSTANTS
      PI = 3.141592653589792
C  PERFORM CONVERSION
      DEG1 = RAD1 * 180.0 / PI
      DEG2 = RAD2 * 180.0 / PI
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
C  CHECK FOR VALIDITY OF SOLUTIONS
      SUBROUTINE VALID(A)
      REAL A(4,7)
      INTEGER I,J
C  CHECK FOR VALIDITY ON ALL JOINTS.IF OUT OF RANGE, SET COLUMN 1 = 1.0
C  NOTE THAT THE RANGES ARE OFFSET BY 0.01 DEGREES TO TAKE CARE
C  OF COMPUTATIONAL ERRORS CAUSED BY THE MACHINE.
      DO 200 I = 1,4
C  WAIST RANGEG IS +/- 147 DEGREES
        IF(ABS(A(I,2) .GT. 147.01) .OR.
C  SHOULDER RANGE IS +56 TO -236 DEGREES
     $     ((A(I,3) .GT. 56.01) .OR. (A(I,3) .LT. -236.01)) .OR.
C  ELBOW RANGE IS THE SAME AS THE SHOULDER RANGE
     $     ((A(I,4) .GT. 56.01) .OR. (A(I,4) .LT. -236.01)) .OR.
C  WRIST ROLL IS CONTINUOUS. RANGE IS +/- 360 DEGREES
     $     ABS(A(I,5) .GT. 360.01) .OR.
C  WRIST PITCH IS +/- 90 DEGREES
     $     ABS(A(I,6) .GT.  90.01) .OR.
C  HAND ROLL IS CONTINUOUS. RANGE IS +/- 360 DEGREES
     $     ABS(A(I,7) .GT. 360.01)) THEN
C  IF OUT OF RANGE, SET FLAG (COLUMN 1 OF RESPECTIVE ROW) = 1.0
          A(I,1) = 1.0
        ENDIF
 200  CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  NORMALIZE THE COMPUTED RESULTS SO THAT EACH ANGLE RANGES
C  FROM -180.0 TO 180.0 DEGREES
      SUBROUTINE NORMAL(A)
      REAL A(4,7)
      INTEGER I,J
C  NORMALIZE THE ANGLES TO BETWEEN -180 AND +180 DEGREES
      DO 701 I = 1,4
        DO 701 J = 2,7
          IF(A(I,J) .GT. 180.0) THEN
            A(I,J) = A(I,J) - 360.0
            ELSEIF(A(I,J) .LT. -180.0) THEN
            A(I,J) = A(I,J) + 360.0
            ENDIF
 701  CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

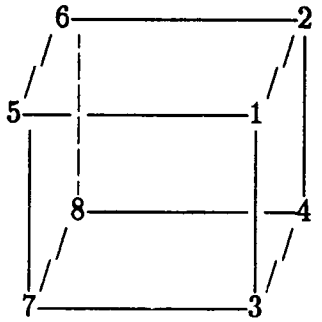
DATAFILE ROOM.DAT

```
2

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

```
144 144 144 144 0 0 0 0
90 90 0 0 90 90 0 0
108 0 108 0 108 0 108 0

0 0 0
0 0 0

-72 -45 -54
0 0 0
```

DOCUMENTED DATA FILE ROOM.DOC

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
WARNING:
     THIS FILE SERVES AS A DOCUMENTATION FOR THE DATA FILE USED TO
     DRAW  THE ROOM THAT WILL CONTAIN THE MERLIN ROBOT.
     IT IS NOT TO BE USED FOR THE ACTUAL DRAWING.
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
     THE FIRST NUMBER TELLS THE HOST PROGRAM HOW MANY LINKS COMPOSE
     THE SYSTEM THAT IS TO BE READ.
2
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EACH LINK IN EACH SYSTEM OF THE CURRENT DRAWING ROUTINE USED IS AN
OBJECT DEFINED BY EIGHT POINTS IN 3D SPACE.


THE FIRST ROW OF NUMBERS IS VALUES OF 'X' FOR THE EIGHT POINTS.
THE SECOND ROW IS FOR THE 'Y' VALUES. THE THIRD ROW IS FOR 'Z' VALUES.


THE ORDER OF THE POINTS IS IMPORTANT.
```



IN THE PROGRAM, THE POINTS ARE CONNECTED IN THE FOLLOWING ORDER

1,2,4,3,7,5,6,2,6,8,4,8,7,5,1,3

```
THIS TRACES OUT THE FIGURE(WHICH NEED NOT BE A CUBE).
SOME LINES ARE RETRACED.
IN DISSPLA, THIS IS THE MOST EFFICIENT METHOD OF DRAWING THE OBJECT.

THE FIRST LINK IN THE DATA FILE IS BLANK, IT IS AN ORIENTATION LINK.
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

THIS IS THE SET OF POINTS THAT DEFINE THE CORNERS OF THE ROOM.
144 144 144 144 0 0 0 0
90 90 0 0 90 90 0 0
108 0 108 0 108 0 108 0

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
THE FOLLOWING NUMBERS DEFINE THE TRANSLATION VECTOR AND ORIENTATION
VECTOR OF A LINK WITH RESPECT TO THE PREVIOUS LINK.
THEY ARE DEFINED AS X,Y,A COORDINATE VALUES.

THE FIRST IS BLANK, IT IS AN ORIENTATION VECTOR.
0 0 0
0 0 0

TLESE VECTORS PLACE THE BACK LEFT BOTTOM CORNER OF THE ROOM WITH
RESPECT TO THE ORIGIN, WHICH IS CENTER OF THE SCREEN.
-72 -45 -54     TRANSLATION VECTOR
0 0 0           ROTATION VECTOR
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

DATAFILE MERL.DAT

```
10

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

18.75 18.75 18.75 18.75 -18.75 -18.75 -18.75 -18.75
4.1 4.1 0 0 4.1 4.1 0 0
18.75 -18.75 18.75 -18.75 18.75 -18.75 18.75 -18.75

-3.12 -3.12 -3.12 -3.12 3.12 3.12 3.12 3.12
-27.9 -27.9 0 0 -27.9 -27.9 0 0
3.12 -3.12 3.12 -3.12 3.12 -3.12 3.12 -3.12

-20.3 -20.3 -20.3 -20.3 7.0 7.0 7.0 7.0
9.0 -9.0 9.0 -9.0 9.0 -9.0 9.0 -9.0
4.9 4.9 -14.4 -14.4 4.9 4.9 -14.4 -14.4

-15 -15 -15 -15 3 3 3 3
-4 -4 4 4 -3 -3 3 3
3 -3 3 -3 3 -3 3 -3
```

```
-3 -3 -3 -3 20.3  20.3  20.3  20.3
-3 -3 3 3 -3 -3 3 3
3 -3 3 -3 3 -3 3 -3

3 3 -3 -3 3 3 -3 -3
-14 -14 -14 -14 17.2 17.2 17.2 17.2
3 -3 3 -3 3 -3 3 -3

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

0 0 0
0 0 0

40 0 54
0 0 0

0 4.1 0
0 0 3.1416

0 -42.35 0
1.5708 0 0

0 12 0
-1.5708 0 0

0 0 6
0 0 0

17.3 0 -6
0 0 -1.5708

0 17.2 0
-1.5708 0 0

0 0 0
1.5708 0 0

0 0 0
-1.5708 0 0
```

DOCUMENTED DATA FILE MERL.DOC

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
WARNING:
THIS FILE SERVES AS A DOCUMENTATION FOR THE DATA FILE USED TO DRAW THE MERLIN ROBOT. IT IS NOT TO BE USED FOR THE ACTUAL DRAWING.

DOCUMENTED DATA FILE:-
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
THE FIRST NUMBER TELLS THE HOST PROGRAM HOW MANY LINKS COMPOSE THE SYSTEM THAT IS TO BE READ.
10

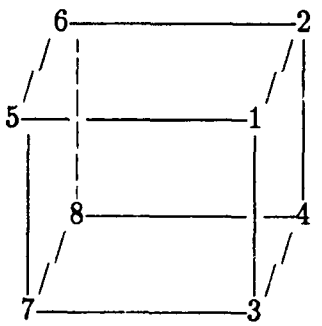
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
THE FIRST LINK IN THE DATA FILE MUST BE A BLANK LINK IF THE SYSTEM IS TO BE APPENDED TO ANOTHER SYSTEM (THE ROBOT IS GOING TO BE PLACED IN THE ROOM, WHICH IS ITS OWN SYSTEM).

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EACH LINK IN EACH SYSTEM OF THE CURRENT DRAWING ROUTINE USED IS AN OBJECT DEFINED BY EIGHT POINTS IN 3D SPACE.

THE FIRST ROW OF NUMBERS IS VALUES OF 'X' FOR THE EIGHT POINTS.
THE SECOND ROW IS FOR THE 'Y' VALUES. THE THIRD ROW IS FOR 'Z' VALUES.

THE ORDER OF THE POINTS IS IMPORTANT.



IN THE PROGRAM, THE POINTS ARE CONNECTED IN THE FOLLOWING ORDER
1,2,4,3,7,5,6,2,6,8,4,8,7,5,1,3
THIS TRACES OUT THE FIGURE(WHICH NEED NOT BE A CUBE). SOME LINES ARE RETRACED. IN DISSPLA, THIS IS THE MOST EFFICIENT METHOD OF DRAWING THE OBJECT.

THIS IS THE SET OF POINTS REPRESENTING THE BASE SUPPORT.
18.75 18.75 18.75 18.75 -18.75 -18.75 -18.75 -18.75
4.1 4.1 0 0 4.1 4.1 0 0
18.75 -18.75 18.75 -18.75 18.75 -18.75 18.75 -18.75

THIS IS THE SET OF POINTS REPRESENTING THE COLUMN BETWEEN THE BASE & THE MOTOR HOUSING.
-3.12 -3.12 -3.12 -3.12 3.12 3.12 3.12 3.12
-27.9 -27.9 0 0 -27.9 -27.9 0 0
3.12 -3.12 3.12 -3.12 3.12 -3.12 3.12 -3.12

THIS IS THE SET OF POINTS REPRESENTING THE WAIST AND MOTOR HOUSING.
-20.3 -20.3 -20.3 -20.3 7.0 7.0 7.0 7.0
9.0 -9.0 9.0 -9.0 9.0 -9.0 9.0 -9.0
4.9 4.9 -14.4 -14.4 4.9 4.9 -14.4 -14.4

THIS IS THE SET OF POINTS REPRESENTING THE COUNTERWEIGHT OF SHOULDER.
-15 -15 -15 -15 3 3 3 3
-4 -4 4 4 -3 -3 3 3
3 -3 3 -3 3 -3 3 -3

THIS IS THE SET OF POINTS REPRESENTING THE UPPER ARM.
-3 -3 -3 -3 20.3 20.3 20.3 20.3
-3 -3 3 3 -3 -3 3 3
3 -3 3 -3 3 -3 3 -3

THIS IS THE SET OF POINTS REPRESENTING THE LOWER ARM.
THE COUNTERWEIGHT IS INCLUDED IN THIS DATA SET.
3 3 -3 -3 3 3 -3 -3
-14 -14 -14 -14 17.2 17.2 17.2 17.2
3 -3 3 -3 3 -3 3 -3

THIS IS THE SET OF POINTS REPRESENTING THE WRIST ROLL. IT IS BLANK BECAUSE IT HAS NO PHYSICAL DIMENSIONS, BUT BECAUSE OF THE WAY THE PROGRAM IS SET UP, THE LINK MUST HAVE DIMENSIONS EVEN IF THEY ARE ZERO.
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

THIS IS THE SET OF POINTS REPRESENTING THE WRIST PITCH.
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

THIS IS THE SET OF POINTS REPRESENTING THE HAND ROLL.
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
THE FOLLOWING NUMBERS DEFINE THE TRANSLATION VECTOR AND ROTATION VECTOR OF A LINK WITH RESPECT TO THE PREVIOUS LINK.

THE FIRST SET MUST BE BLANK SO THAT THE MERLIN CAN BE APPENDED TO THE ROOM.

THE ZEROES WILL BE REPLACED IN THE PROGRAM WITH THE VECTORS OF THE LAST LINK OF THE SYSTEM TO BE APPENDED TO.

```
0 0 0  - TRANSLATION VECTOR
0 0 0  - ROTATION VECTOR

THIS IS THE TRANSLATION VECTOR OF THE BASE.
THEY ARE DEFINED AS X,Y,Z COORDINATES OF THE BOTTOM CENTER OF THE BASE
WITH RESPECT TO THE FAR LEFT LOWER CORNER OF THE ROOM.
IN THE PROGRAM, WHEN THE ROBOT IS TO BE REPOSITIONED, THESE NUMBERS WILL
BE CHANGED BY THE USER. THE DEFAULT IS SET TO [54,0,54]

THESE ARE THE VECTORS OF THE ROBOT'S BASE WITH RESPECT TO THE ROOM.
54 0 54    - TRANSLATION VECTOR
0 0 0      - ROTATION VECTOR

THESE ARE THE VECTORS OF THE ROBOT'S COLUMN WITH RESPECT TO THE BASE.
0 4.1 0          - TRANSLATION VECTOR
0 0 3.1416       - ROTATION VECTOR

THESE ARE THE VECTORS OF THE ROBOT'S WAIST WITH RESPECT TO THE COLUMN.
0 -42.35 0       - TRANSLATION VECTOR
1.5708 0 0       - ROTATION VECTOR

THESE ARE THE VECTORS OF THE ROBOT'S COUNTERWEIGHT WITH RESPECT TO THE
WAIST.
0 12 0           - TRANSLATION VECTOR
-1.5708 0 0      - ROTATION VECTOR

THESE ARE THE VECTORS OF THE ROBOT'S UPPER ARM WITH RESPECT TO THE
SHOULDER COUNTERWEIGHT.
0 0 6            - TRANSLATION VECTOR
0 0 0            - ROTATION VECTOR

THESE ARE THE VECTORS OF THE ROBOT'S LOWER ARM WITH RESPECT TO THE UPPER
ARM.
17.3 0 -6        - TRANSLATION VECTOR
0 0 -1  708      - ROTATION VECTOR

THESE ARE THE VECTORS OF THE ROBOT'S WRIST ROLL WITH RESPECT TO THE
LOWER ARM.
0 17.2 0         - TRANSLATION VECTOR
-1.5708 0 0      - ROTATION VECTOR

THESE ARE THE VECTORS OF THE ROBOT'S WRIST PITCH WITH RESPECT TO THE
WRIST ROLL.
0 0 0            - TRANSLATION VECTOR
1.5708 0 0       - ROTATION VECTOR

THESE ARE THE VECTORS OF THE ROBOT'S HAND ROLL WITH RESPECT TO THE WRIST
PITCH.
0 0 0            - TRANSLATION VECTOR
-1.5708 0 0      - ROTATION VECTOR
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

DATAFILE UTAH.DAT

```
4


0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

1.8 -1.9 1.8 -1.9 1.8 -1.9 1.8 -1.9
4.25 4.25 4.25 4.25 0 0 0 0
.375 .375 -1.5 -1.5 .375 .375 -.375 -.375

0 0 0
0 0 0

0 0 0
0 0 0

0 0 0
-1.5708 0 3.1416

0 0 0
0 1.5708 0

5

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

-.5 -.5 .5 .5 -.5 -.5 .5 .5
-.5 .5 -.5 .5 -.5 .5 -.5 .5
1.2 1.2 1.2 1.2 0 0 0 0

1.7 1.7 1.7 1.7 0 0 0 0
.45 .45 -.45 -.45 .45 .45 -.45 -.45
.45 -.45 .45 -.45 .45 -.45 .45 -.45

1.3 1.3 1.3 1.3 0 0 0 0
.4 .4 -.4 -.4 .4 .4 -.4 -.4
.4 -.4 .4 -.4 .4 -.4 .4 -.4
1.0625 1.0625 1.0625 1.0625 0 0 0 0
.35 .35 -.35 -.35 .35 .35 -.35 -.35
.35 -.35 .35 -.35 .35 -.35 .35 -.35
```

```
0 0 0
0 0 0

-1.375 4.25 -.75
-1.34 0 1.5708

0 0 1.2
1.5708 0 1.5708
1.7 0 0
0 0 0

1.3 0 0
0 0 0

5

0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0

-.5 -.5 .5 .5 -.5 -.5 .5 .5
-.5 .5 -.5 .5 -.5 .5 -.5 .5
1.2 1.2 1.2 1.2 0 0 0 0

1.7 1.7 1.7 1.7 0 0 0 0
.45 .45 -.45 -.45 .45 .45 -.45 -.45
.45 -.45 .45 -.45 .45 -.45 .45 -.45

1.3 1.3 1.3 1.3 0 0 0 0
.4 .4 -.4 -.4 .4 .4 -.4 -.4
.4 -.4 .4 -.4 .4 -.4 .4 -.4

1.0625 1.0625 1.0625 1.0625 0 0 0 0
.35 .35 -.35 -.35 .35 .35 -.35 -.35
.35 -.35 .35 -.35 .35 -.35 .35 -.35

0 0 0
0 0 0

 .10 4.25 -.75
-1.34 0 1.5708

0 0 1.2
1.5708 0 1.5708

1.7 0 0
0 0 0
1.3 0 0
0 0 0

5
```

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

-.5 -.5 .5 .5 -.5 -.5 .5 .5
-.5 .5 -.5 .5 -.5 .5 -.5 .5
1.2 1.2 1.2 1.2 0 0 0 0

1.7 1.7 1.7 1.7 0 0 0 0
.45 .45 -.45 -.45 .45 .45 -.45 -.45
.45 -.45 .45 -.45 .45 -.45 .45 -.45

1.3 1.3 1.3 1.3 0 0 0 0
.4 .4 -.4 -.4 .4 .4 -.4 -.4
.4 -.4 .4 -.4 .4 -.4 .4 -.4

1.0625 1.0625 1.0625 1.0625 0 0 0 0
.35 .35 -.35 -.35 .35 .35 -.35 -.35
.35 -.35 .35 -.35 .35 -.35 .35 -.35

0 0 0
0 0 0

1.1875 4.25 -.75
-1.34 0 1.5708

0 0 1.2
1.5708 0 1.5708

1.7 0 0
0 0 0

1.3 0 0
0 0 0

5

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

1.7 1.7 0 0 1.7 1.7 0 0
.5 .5 .5 .5 -.5 -.5 -.5 -.5
-.5 .5 -.5 .5 -.5 .5 -.5 .5

1.3125 1.3125 0 0 1.3125 1.3125 0 0
.45 .45 .45 .45 -.45 -.45 -.45 -.45
-.45 .45 -.45 .45 -.45 .45 -.45 .45
```

```
1.125 1.125 0 0 1.125 1.125 0 0
.4 .4 .4 .4 -.4 -.4 -.4 -.4
-.4 .4 -.4 .4 -.4 .4 -.4 .4

0 0 0
0 0 0

.75 3.125 0
-1.5708 0 -1.5708

.375 0 0
1.5708 0 0

1.7 0 0
0 0 0

1.3125 0 0
0 0 0



.375 .375 -.375 -.375 .375 .375 -.375 -.375
3.875 3.875 3.875 3.875 0 0 0 0
-1.9 1.8 -1.9 1.8 -1.9 1.8 -1.9 1.8
```

DOCUMENTED DATA FILE UTAH.DOC

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
WARNING:
THIS FILE SERVES AS A DOCUMENTATION FOR THE DATA FILE USED TO DRAW
THE UTAH HAND. IT IS NOT TO BE USED FOR THE ACTUAL DRAWING.

DOCUMENTED DATA FILE:-
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
     THE FIRST NUMBER TELLS THE HOST PROGRAM HOW MANY LINKS COMPOSE
     THE SYSTEM THAT IS TO BE READ.
4

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
THE FIRST LINK IN THE DATA FILE OF SYSTEM MUST BE A BLANK LINK IF THE
SYSTEM IS TO BE APPENDED TO ANOTHER SYSTEM(THE HAND IS GOING TO BE
PLACED ON THE MERLIN ROBOT).

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
THE SECOND LINK IN THE DATA FILE FOR THE HAND MUST BE A BLANK.
IN THE PROGRAM, THE USER HAS THE OPTION OF REPOSITIONING THE HAND.
```

TO MAKE THIS POSSIBLE, A BLANK LINK IS REQUIRED.

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

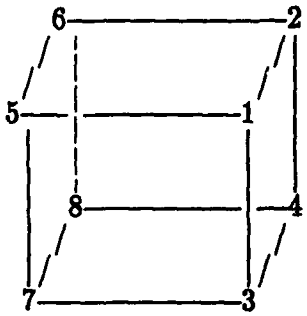
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
THE THIRD LINK IN THE DATA FILE FOR THE HAND MUST BE A BLANK. THIS IS NECESSARY TO ACHIEVE PROPER ORIENTATION OF THE HAND.

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EACH LINK IN EACH SYSTEM OF THE CURRENT DRAWING ROUTINE USED IS AN OBJECT DEFINED BY EIGHT POINTS IN 3D SPACE.

THE FIRST ROW OF NUMBERS IS VALUES OF 'X' FOR THE EIGHT POINTS.
THE SECOND ROW IS FOR THE 'Y' VALUES. THE THIRD ROW IS FOR 'Z' VALUES.

THE ORDER OF THE POINTS IS IMPORTANT.



IN THE PROGRAM, THE POINTS ARE CONNECTED IN THE FOLLOWING ORDER
1,2,4,3,7,5,6,2,6,8,4,8,7,5,1,3
THIS TRACES OUT THE FIGURE(WHICH NEED NOT BE A CUBE). SOME LINES ARE RETRACED. IN DISSPLA, THIS IS THE MOST EFFICIENT METHOD OF DRAWING THE OBJECT.

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
THIS IS THE SET OF POINTS REPRESENTING THE PALM SECTION.
1.8 -1.9 1.8 -1.9 1.8 -1.9 1.8 -1.9
4.25 4.25 4.25 4.25 0 0 0 0
.375 .375 -1.5 -1.5 .375 .375 -.375 -.375

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
THE FOLLOWING NUMBERS DEFINE THE TRANSLATION VECTOR AND ROTATION VECTOR OF THAT LINK WITH RESPECT TO THE PREVIOUS LINK. THEY ARE DEFINED AS X,Y,Z COORDINATE VALUES.

THE FIRST MUST BE BLANK IF IT IS TO APPENDED TO ANOTHER SYSTEM
THE ZEROES WILL BE REPLACED IN THE PROGRAM WITH THE VECTORS OF THE LAST

```
LINK OF THE SYSTEM TO BE APPENDED TO.
0 0 0           TRANSLATION VECTOR
0 0 0           ROTATION VECTOR

THE SECOND MUST BE BLANK TO ALLOW FOR REPOSITION OF THE HAND.
THE ZEROES WILL BE REPLACED IN THE PROGRAM WITH THE VECTORS THAT DEFINE
THE REPOSITIONING OF THE HAND.
0 0 0           TRANSLATION VECTOR
0 0 0           ROTATION VECTOR


THESE ARE THE TRANSLATION AND ROTATION VECTORS OF SECOND AND THIRD EMPTY
LINKS.
0 0 0           TRANSLATION VECTOR
-1.57 0 3.14    ROTATION VECTOR

THESE ARE VECTORS OF THE PALM WITH RESPECT TO THE THIRD EMPTY LINK.
0 0 0           TRANSLATION VECTOR
0 1.5708 0      ROTATION VECTOR


CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
NUMBER OF LINKS OF THE FIRST FINGER.
5

EMPTY LINK FOR APPENDING THE FIRST FINGER TO THE PALM.
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

THIS IS THE SET OF POINTS REPRESENTING THE (ZERO)TH LINK SECTION.
-.5 -.5 .5 .5 -.5 -.5 .5 .5
-.5 .5 -.5 .5 -.5 .5 -.5 .5
1.2 1.2 1.2 1.2 0 0 0 0

THIS IS THE SET OF POINTS REPRESENTING THE FIRST LINK SECTION.
1.7 1.7 1.7 1.7 0 0 0 0
.45 .45 -.45 -.45 .45 .45 -.45 -.45
.45 -.45 .45 -.45 .45 -.45 .45 -.45

THIS IS THE SET OF POINTS REPRESENTING THE SECOND LINK SECTION.
1.3 1.3 1.3 1.3 0 0 0 0
.4 .4 -.4 -.4 .4 .4 -.4 -.4
.4 -.4 .4 -.4 .4 -.4 .4 -.4

THIS IS THE SET OF POINTS REPRESENTING THE THIRD LINK SECTION.
1.0625 1.0625 1.0625 1.0625 0 0 0 0
.35 .35 -.35 -.35 .35 .35 -.35 -.35
.35 -.35 .35 -.35 .35 -.35 .35 -.35

EMPTY TRANSLATION AND ROTATION VECTORS FOR APPENDING THE FINGER TO THE
HAND.
0 0 0
0 0 0
```

```
THESE ARE THE VECTORS OF THE ZEROTH LINK WITH RESPECT TO THE HAND.
-1.375 4.25 -.75     TRANSLATION VECTOR
-1.34 0 1.5708       ROTATION VECTOR

THESE ARE THE VECTORS OF THE FIRST LINK WITH RESPECT TO THE (ZERO)TH.
0 0 1.2              TRANSLATION VECTOR
1.5708 0 1.5708      ROTATION VECTOR

THESE ARE THE VECTORS OF THE SECOND LINK WITH RESPECT TO THE FIRST.
1.7 0 0              TRANSLATION VECTOR
0 0 0                ROTATION VECTOR

THESE ARE THE VECTORS OF THE THIRD LINK WITH RESPECT TO THE SECOND.
1.3 0 0              TRANSLATION VECTOR
0 0 0                ROTATION VECTOR


CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
NUMBER OF LINKS OF THE SECOND FINGER.
5

EMPTY LINK FOR APPENDING THE SECOND FINGER TO THE PALM.
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

THIS IS THE SET OF POINTS REPRESENTING THE (ZERO)TH LINK SECTION.
-.5 -.5 .5 .5 -.5 -.5 .5 .5
-.5 .5 -.5 .5 -.5 .5 -.5 .5
1.2 1.2 1.2 1.2 0 0 0 0

THIS IS THE SET OF POINTS REPRESENTING THE FIRST LINK SECTION.
1.7 1.7 1.7 1.7 0 0 0 0
.45 .45 -.45 -.45 .45 .45 -.45 -.45
.45 -.45 .45 -.45 .45 -.45 .45 -.45

THIS IS THE SET OF POINTS REPRESENTING THE SECOND LINK SECTION.
1.3 1.3 1.3 1.3 0 0 0 0
.4 .4 -.4 -.4 .4 .4 -.4 -.4
.4 -.4 .4 -.4 .4 -.4 .4 -.4

THIS IS THE SET OF POINTS REPRESENTING THE THIRD LINK SECTION.
1.0625 1.0625 1.0625 1.0625 0 0 0 0
.35 .35 -.35 -.35 .35 .35 -.35 -.35
.35 -.35 .35 -.35 .35 -.35 .35 -.35

EMPTY TRANSLATION AND ROTATION VECTORS FOR APPENDING THE FINGER TO THE
HAND.
0 0 0
0 0 0

THESE ARE THE VECTORS OF THE (ZERO)TH LINK WITH RESPECT TO THE HAND.
-.10 4.25 -.75       TRANSLATION VECTOR
-1.34 0 1.5708       ROTATION VECTOR
```

```
THESE ARE THE VECTORS OF THE FIRST LINK WITH RESPECT TO THE (ZERO)TH.
0 0 1.2              TRANSLATION VECTOR
1.5708 0 1.5708      ROTATION VECTOR

THESE ARE THE VECTORS OF THE SECOND LINK WITH RESPECT TO THE FIRST.
1.7 0 0              TRANSLATION VECTOR
0 0 0                ROTATION VECTOR

THESE ARE THE VECTORS OF THE THIRD LINK WITH RESPECT TO THE SECOND.
1.3 0 0              TRANSLATION VECTOR
0 0 0                ROTATION VECTOR


CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
NUMBER OF LINKS OF THE THIRD FINGER.
5

EMPTY LINK FOR APPENDING THE THIRD FINGER TO THE PALM.
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

THIS IS THE SET OF POINTS REPRESENTING THE (ZERO)TH LINK SECTION.
-.5 -.5 .5 .5 -.5 -.5 .5 .5
-.5 .5 -.5 .5 -.5 .5 -.5 .5
1.2 1.2 1.2 1.2 0 0 0 0

THIS IS THE SET OF POINTS REPRESENTING THE FIRST LINK SECTION.
1.7 1.7 1.7 1.7 0 0 0 0
.45 .45 -.45 -.45 .45 .45 -.45 -.45
.45 -.45 .45 -.45 .45 -.45 .45 -.45

THIS IS THE SET OF POINTS REPRESENTING THE SECOND LINK SECTION.
1.3 1.3 1.3 1.3 0 0 0 0
.4 .4 -.4 -.4 .4 .4 -.4 -.4
.4 -.4 .4 -.4 .4 -.4 .4 -.4

THIS IS THE SET OF POINTS REPRESENTING THE THIRD LINK SECTION.
1.0625 1.0625 1.0625 1.0625 0 0 0 0
.35 .35 -.35 -.35 .35 .35 -.35 -.35
.35 -.35 .35 -.35 .35 -.35 .35 -.35

EMPTY TRANSLATION AND ROTATION VECTORS FOR APPENDING THE FINGER TO THE
HAND.
0 0 0
0 0 0

THESE ARE THE VECTORS OF THE (ZERO)TH LINK WITH RESPECT TO THE HAND.
1.1875 4.25 -.75     TRANSLATION VECTOR
-1.34 0 1.5708       ROTATION VECTOR

THESE ARE THE VECTORS OF THE FIRST LINK WITH RESPECT TO THE (ZERO)TH.
0 0 1.2              TRANSLATION VECTOR
1.5708 0 1.5708      ROTATION VECTOR
```

```
THESE ARE THE VECTORS OF THE SECOND LINK WITH RESPECT TO THE FIRST.
1.7 0 0              TRANSLATION VECTOR
0 0 0                ROTATION VECTOR

THESE ARE THE VECTORS OF THE THIRD LINK WITH RESPECT TO THE SECOND.
1.3 0 0              TRANSLATION VECTOR
0 0 0                ROTATION VECTOR


CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
NUMBER OF LINKS OF THE THUMB.
5

EMPTY LINK FOR APPENDING THE THUMB TO THE PALM.
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

THIS IS THE SET OF POINTS REPRESENTING THE (ZERO)TH LINK SECTION.
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

THIS IS THE SET OF POINTS REPRESENTING THE FIRST LINK SECTION.
1.7 1.7 0 0 1.7 1.7 0 0
.5 .5 .5 .5 -.5 -.5 -.5 -.5
-.5 .5 -.5 .5 -.5 .5 -.5 .5

THIS IS THE SET OF POINTS REPRESENTING THE SECOND LINK SECTION.
1.3125 1.3125 0 0 1.3125 1.3125 0 0
.45 .45 .45 .45 -.45 -.45 -.45 -.45
-.45 .45 -.45 .45 -.45 .45 -.45 .45

THIS IS THE SET OF POINTS REPRESENTING THE THIRD LINK SECTION.
1.125 1.125 0 0 1.125 1.125 0 0
.4 .4 .4 .4 -.4 -.4 -.4 -.4
-.4 .4 -.4 .4 -.4 .4 -.4 .4

EMPTY TRANSLATION AND ROTATION VECTORS FOR APPENDING THE THUMB TO THE
HAND.
0 0 0
0 0 0

THESE ARE THE VECTORS OF THE (ZERO)TH LINK WITH RESPECT TO THE HAND.
.75 3.125 0          TRANSLATION VECTOR
-1.5708 0 -1.5708    ROTATION VECTOR

THESE ARE THE VECTORS OF THE FIRST LINK WITH RESPECT TO THE (ZERO)TH
LINK.
.375 0 0         TRANSLATION VECTOR
1.5708 0 0       ROTATION VECTOR
```

```
THESE ARE THE VECTORS OF THE SECOND LINK WITH RESPECT TO THE FIRST.
1.7 0 0          TRANSLATION VECTOR
0 0 0            ROTATION VECTOR

THESE ARE THE VECTORS OF THE THIRD LINK WITH RESPECT TO THE SECOND.
1.3125 0 0       TRANSLATION VECTOR
0 0 0            ROTATION VECTOR
```

REFERENCES

1) Robert D. Ballard, "A Long Last Look at *Titanic*", National Geographic, Volume 170, Number 6, pp. 698-727. December 1986.

2) C. R. Weisbin, "Robotics and Intelligent Systems Program", Informational report published by the Oak Ridge National Laboratory, Oak Ridge, Tennessee. 1987

3) Ichiro Kato, Kuni Sadamoto, "Mechanical Hands Illustrated", Hemisphere Publishing Corp. 1987.

4) Jean Vertut, Philippe Coiffet, "Teleoperation and Robotics - Evolution and Development", Volumes 3A and 3B. Prentice Hall, Inc. 1985.

5) John J. Craig, "Introduction to Robotics - Mechanisms and Control", Addison-Wesley Publishing Co., Reading, Massachusetts. 1986.

6) Bernard Roth, "Performance Evaluation of Manipulators from a Kinematic Viewpoint", National Bureau of Standards Workshop on Performance Evaluation of Manipulators, Annapolis, Maryland. October 23 - 25, 1975.

7) J. Denavit, R. S. Hartenburg, "A Kinematic Notation for Lower-Pair Mechanisms based on Matrices", ASME Journal of Applied Mechanics, Vol 22(2), pp. 215 - 221. 1955.

8) Merlin^{tm} System Operators Guide, Version 3.0, American Robot Corp. June 1985.

9) A. Kumar, K. J. Waldron, "The Workspaces of a Mechanical Manipulator", Journal of Mechanical Design, Volume 103, pp. 665 - 672. July 1981.

10) J. A. Hansen, K. C. Gupta, S. M. K. Kazerounian, "Generation and Evaluation of the Workspace of a Manipulator", The International Journal of Robotics Research, Volume 2, No. 3. Fall 1983.

11) T. W. Lee, D. C. H. Yang, "On the Evaluation of Manipulator Workspaces", Transactions of the ASME - Journal of Mechanisms, Transmissions and Automation Design, Vol. 105. March 1983.

12) J. K. Salisbury, "Kinematics and Force Analysis of Articulated Hands", Ph. D. Thesis, Department of Mechanical Engineering, Stanford University. May 1982.

13) S. Narasimhan, "Dexterous Robotic Hands: Kinematics and Control", M.S. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. January 1988.

14) S. C. Jacobsen, J. E. Wood, D. F. Knutti, K. B. Biggers, "The UTAH/MIT Dexterous Hand: Work in Progress", International Journal

of Robotics Research, Volume 3, No. 4, pp. 21 - 50. Winter 1984.

15) S. C. Jacobsen, E. K. Iversen, D. F. Knutti, R. T. Johnson, K. B. Biggers, "Design of the UTAH/MIT Dexterous Hand", Proc. IEEE International Conference on Robotics and Automation, San Francisco, California. April 7 - 10, 1986

*U.S.Government Printing Office: 1989 — 648-055/02007